DISUSUN OLEH : ADAM MUKHARIL BACHTIAR, S.Kom.

BAB III
TIPE DATA, VARIABEL, DAN OPERASI PERHITUNGAN

IF

Tipe Data

Pembuatan sebuah program dimaksudkan untuk mengolah data masukan. Setiap data yang akan diolah di dalam bahasa C dan bahasa C++ harus jelas tipe datanya. Tipe data pada bahasa C dan C++ secara umum terbagi menjadi tiga macam, yaitu:

1. Tipe data karakter
2. Tipe data bilangan bulat
3. Tipe data bilangan pecahan.

Apabila masih dirasa kurang, seorang programmer dapat mendeklarasikan tipe data baru dengan menggunakan perintah **struct**. Untuk perintah struct akan dijelaskan pada bab berikutnya. Yang harus diperhatikan dalam memilih data adalah jangkauan nilainya karena apabila salah dalam memilih tipe data maka perhitungan yang dilakukan menjadi tidak benar.

Ukuran memori yang diperlukan untuk masing-masing tipe data sangat bergantung pada perangkat keras dan perangkat lunak C/C++ yaqng digunakan. Karena itu jangkauan bilangan dari masing-masing tipe data juga bisa berlainan.

Tipe Data Karakter

Tipe data ini digunakan untuk merepresentasikan data-data yang berupa karakter. Tipe data ini dinyatakan dengan tipe **char**. Tipe data ini mempunyai jangkauan dari 0 sampai 255 atau kaakter ASCII ke 0 sampai karakter ASCII 255. Tipe data ini bisa ditampilkan dengan suatu karakter atau suatu bilangan. Tetapi apabila ingin diisi dengan angka maka angka tersebut harus diberi tanda kutip. Dalam bahasa C, tipe data karakter mempunyai format %c, maksudnya apabila ingin memanggil tipe data karakter di dalam fungsi printf harus mencantumkan %c. apabila dipanggil dengan format %i maka yang tampil adalah karakter ASCII dari data tersebut.

Adapun tipe data dalam kategori ini yang didefinisikan oleh ANSI/ISO C++ Standard adalah sebagai berikut:

Tabel 3.1 Tabel tipe data karakter

Tipe Data	Memori (Dalam Byte)	Rentang
char	1	-128 sampai 127 atau 0 sampai 255
unsigned char	1	0 sampai 255
signed char	1	-128 sampai 127

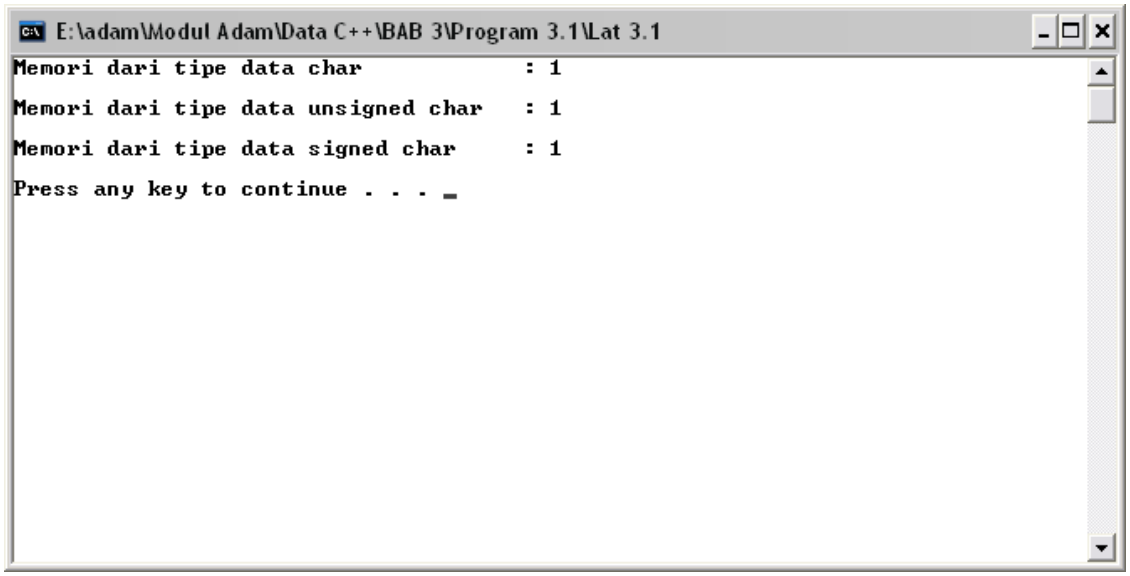
Untuk membuktikan memori yang digunakan per masing-masing tipe data, buatlah program dengan sintaks di bawah ini:

```
1  /*
2    Program 3.1
3    Nama File  : Lat-3.1.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12   printf("Memori dari tipe data char:
            %i\n",sizeof(char));
13   printf("\nMemori dari tipe data unsigned char:
             %i\n",sizeof(unsigned char));
14   printf("\nMemori dari tipe data char:
             %i\n\n",sizeof(signed char));
15   system("PAUSE");
16   return 0;
17 }
```

```
1  /*
2    Program 3.1
3    Nama File  : Lat-3.1.cpp
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
```

```
10  using namespace std;
11  int main(int argc, char *argv[])
12  {
13      cout<<"Memori dari tipe data char            :"
            <<sizeof(char);
14      cout<<"\n\n";
15      cout<<"Memori dari tipe data unsigned char   :"
            <<sizeof(unsigned char);
16      cout<<"\n\n";
17      cout<<"Memori dari tipe data signed char     :"
            <<sizeof(signed char);
18      cout<<"\n\n";
19      system("PAUSE");
20      return EXIT_SUCCESS;
21  }
```

Maka hasil eksekusinya adalah sebagai berikut:



Gambar 3.1 Hasil eksekusi program Lat 3.1

Pemberian nilai pada suatu karakter dilakukan dengan menuliskan perintah sebagai berikut:

1. Karakter='A';
2. Karakter=65;

Kedua cara itu memiliki hasil yang sama yaitu memberikan nilai 65 atau karakter A ke variabel bernama Karakter. Ingat untuk membubuhkan tanda apostrof (‘) apabila ingin mengisi suatu variabel dengan data bertipe karakter. Untuk lebih jelasnya buatlah program dengan sintaks seperti di bawah ini:

```
1   /*
2     Program 3.2
3     Nama File  : Lat-3.2.c
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <stdio.h>
8   #include <stdlib.h>
9
10  int main(int argc, char *argv[])
11  {
12    char A1,A2;
13    A1='A';
14    A2=A1;
15    printf("Nilai variabel A1 adalah %c\n",A1);
16    printf("Nilai variabel A2 dalam bentuk angka (ASCII) =
            %i\n\n",A2);
17    system("PAUSE");
18    return 0;
19  }
```

Penjelasan:

1. char A1,A2;

   Baris perintah ini adalah untuk mendeklarasikan variabel bernama A1 dan A2. Maka setelah baris ini dieksekusi, terbentuklah dua buah variabel bernama A1 dan A2 bertipe data karakter.

2. %i

   %i di sini adalah format dari tipe data integer. Apabila sebuah variabel berisi data karakter dan dipanggil menggunakan format %i maka hasilnya adalah angka ASCII dari variabel tersebut.

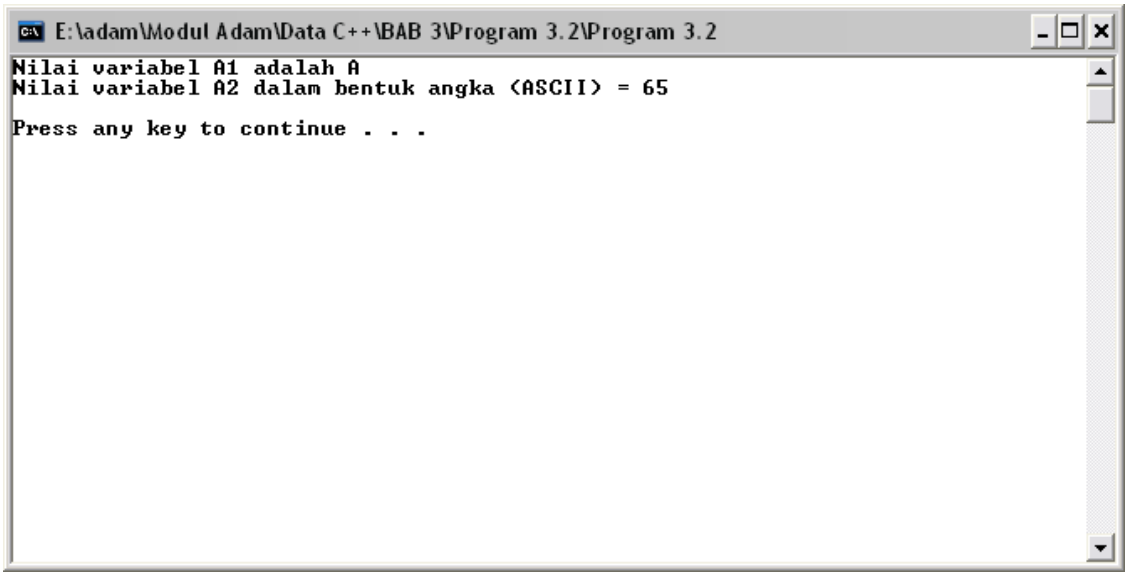
```
1   /*
2     Program 3.2
3     Nama File  : Lat-3.2.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
```

```
12  int main(int argc, char *argv[])
13  {
14      char A1;
15      int A2;
16      A1='A';
17      A2=int(A1);
18      cout<<"Nilai variabel A1 adalah "<<A1;
19      cout<<endl;
20      cout<<"Nilai variabel A2 dalam bentuk angka (ASCII)
              = "<<A2;
21      cout<<"\n\n";
22      system("PAUSE");
23      return EXIT_SUCCESS;
24  }
```

Penjelasan:

Di dalam sintaks di atas terdapat suatu baris perintah yang berisi A2=int(A1);. Baris perintah tersebut digunakan untuk mengkonversi tipe data asal suatu data (dalam kasus ini bertipe karakter) menjadi tipe data integer. Hal ini dilakukan karena dalam bahasa C++ tidak dikenal adanya format data. Mungkin untuk sekarang hal ini terlihat menyulitkan karena harus menambah satu baris perintah tetapi apabila program yang dibuat mulai kompleks maka bahasa C++ akan mempermudah programmernya karena tidak memerlukan format data.

Setelah dijalankan maka hasilnya adalah sebagai berikut:



Gambar 3.2 Hasil eksekusi program Lat 3.2

Tipe Data Bilangan Bulat

Tipe data ini digunakan untuk data-data angka yang tidak mengandung angka di belakang koma. Sebagai contoh 23, 5, 6, dan 1986. Tipe data yang termasuk dalam kategori ini dapat dilihat dalam tabel berikut:

Tabel 3.2 Tabel tipe data bilangan bulat

Tipe Data	Memori (Dalam Byte)	Format	Rentang
int	2 atau 4	%d atau %i	-32.768 sampai 32.767 atau -2.147.483.648 sampai 2.147.483.647
unsigned int	2 atau 4	%u	0 sampai 65.535 atau 0 sampai 4.294.967.295
signed int	2 atau 4	%i	Sama seperti **int**
short int	2	%i	-32.768 sampai 32.767
unsigned short int	2	%u	0 sampai 65.535
signed short int	2	%i	Sama seperti **short int**
long int	4	%ld atau %li	-2.147.483.648 sampai 2.147.483.647
signed long int	4	%li	Sama seperti **long int**
unsigned long int	4	%lu	0 sampai 4.294.967.295

Untuk membuktikan memori yang digunakan per masing-masing tipe data, buatlah program dengan sintaks di bawah ini:

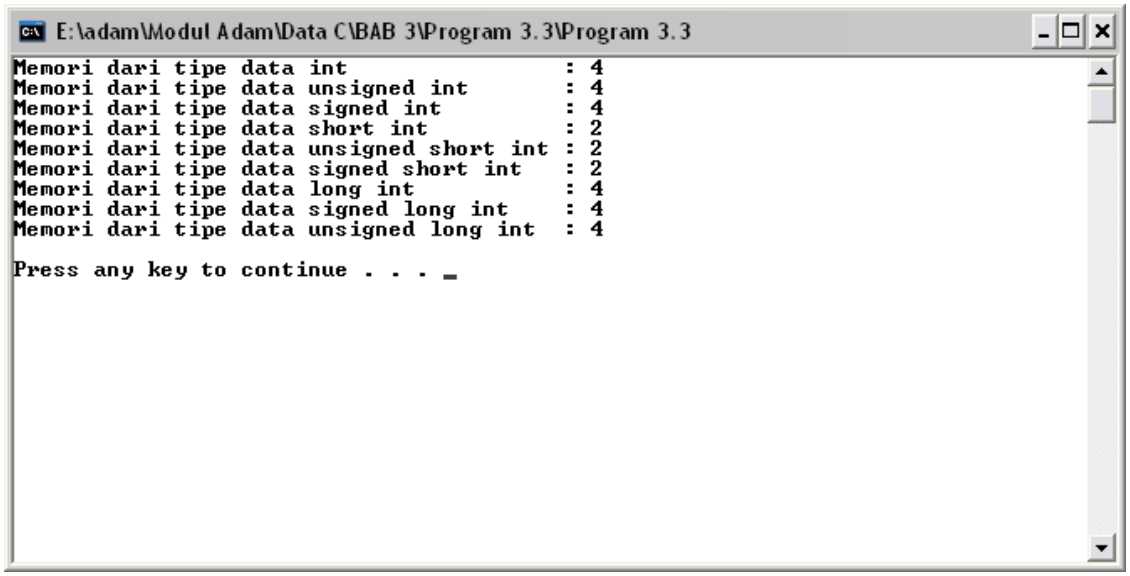
```
1  /*
2    Program 3.3
3    Nama File  : Lat-3.3.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
```

```
12    printf("Memori dari tipe data int                :
13            %i\n",sizeof(int));
14    printf("Memori dari tipe data unsigned int       :
            %i\n",sizeof(unsigned int));
15    printf("Memori dari tipe data signed int         :
            %i\n",sizeof(signed int));
16    printf("Memori dari tipe data short int          :
            %i\n",sizeof(short int));
17    printf("Memori dari tipe data unsigned short int :
            %i\n",sizeof(unsigned short int));
18    printf("Memori dari tipe data signed short int   :
            %i\n",sizeof(signed short int));
19    printf("Memori dari tipe data long int           :
            %i\n",sizeof(long int));
20    printf("Memori dari tipe data signed long int    :
            %i\n",sizeof(signed long int));
21    printf("Memori dari tipe data unsigned long int  :
            %i\n\n",sizeof(unsigned long int));
22    system("PAUSE");
23    return 0;
24  }
```

```
1   /*
2     Program 3.3
3     Nama File  : Lat-3.3.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
12  int main(int argc, char *argv[])
13  {
14      cout<<"Memori dari tipe data int                :
            "<<sizeof(int);
15      cout<<endl;
16      cout<<"Memori dari tipe data unsigned int       :
            "<<sizeof(unsigned int);
17      cout<<endl;
18      cout<<"Memori dari tipe data signed int         :
            "<<sizeof(signed int);
19      cout<<endl;
20      cout<<"Memori dari tipe data short int          :
            "<<sizeof(short int);
21      cout<<endl;
```

```
22      cout<<"Memori dari tipe data unsigned short int :
           "<<sizeof(unsigned short int);
23      cout<<endl;
24      cout<<"Memori dari tipe data signed short int   :
           "<<sizeof(signed short int);
25      cout<<endl;
26      cout<<"Memori dari tipe data long int           :
           "<<sizeof(long int);
27      cout<<endl;
28      cout<<"Memori dari tipe data signed long int    :
           "<<sizeof(signed long int);
29      cout<<endl;
30      cout<<"Memori dari tipe data unsigned long int  :
           "<<sizeof(unsigned long int);
31      cout<<"\n\n";
32      system("PAUSE");
33      return EXIT_SUCCESS;
34  }
```

Setelah dijalankan maka hasilnya adalah sebagai berikut:



Gambar 3.3 Hasil eksekusi program Lat 3.3

Adapun contoh cara pendeklarasian tipe data bilangan bulat antara lain:

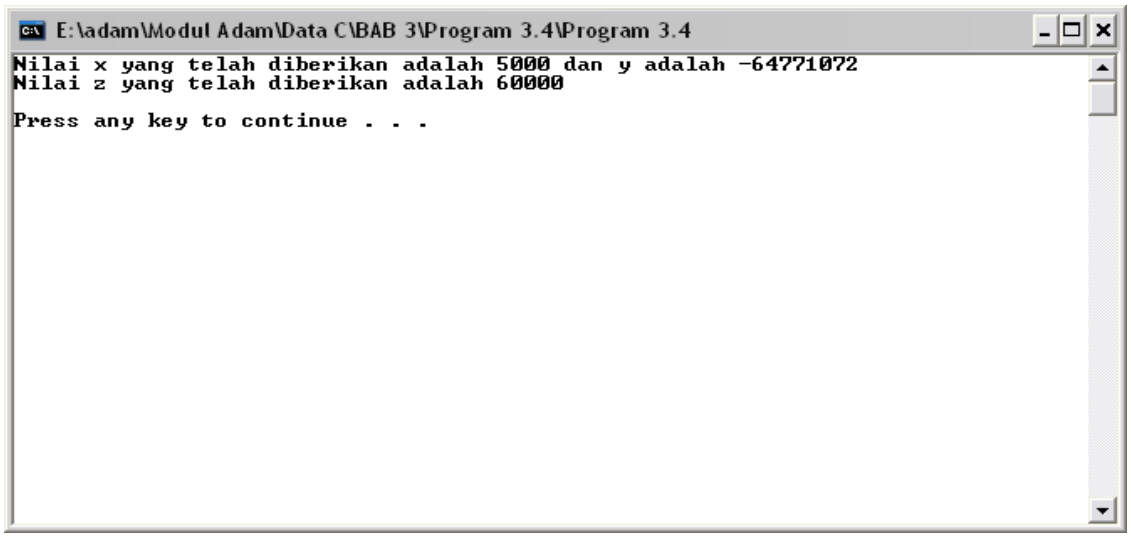
1. Int x,y,z;
2. Unsigned int m;
3. Unsigned short int n;

Catatan	Perhatikan ukuran memori dan jangkauan nilai per masing-masing tipe data untuk mendapatkan hasil perhitungan yang benar. Salah memilih tipe data maka perhitungan yang dilakukan menjadi tidak valid.

Berikut ini adalah contoh program dari tipe data bilangan bulat:

```
1   /*
2     Program 3.4
3     Nama File  : Lat-3.4.c
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <stdio.h>
8   #include <stdlib.h>
9
10  int main(int argc, char *argv[])
11  {
12    int x=5000,y=30000000000;
13    unsigned int z=60000;
14    printf("Nilai x yang telah diberikan adalah %i dan y
            adalah %i\n",x,y);
15    printf("Nilai z yang telah diberikan adalah
            %u\n\n",z);
16    system("PAUSE");
17    return 0;
18  }
```

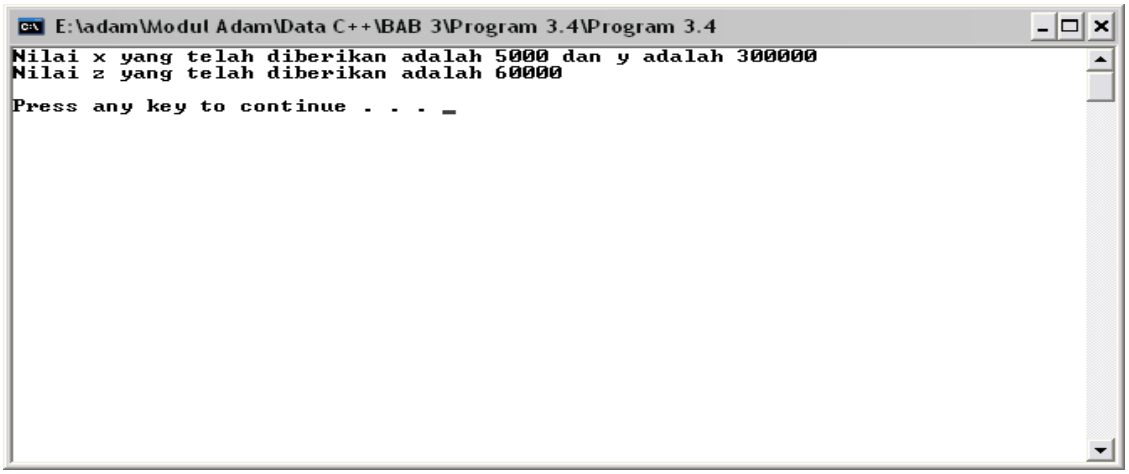
Hasil eksekusi:



Gambar 3.4 Hasil eksekusi program 3.4 dalam bahasa C

```
1   /*
2     Program 3.4
3     Nama File  : Lat-3.4.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
12  int main(int argc, char *argv[])
13  {
14      int x=5000,y=300000;
15      unsigned int z=60000;
16      cout<<"Nilai x yang telah diberikan adalah "<<x<<"
            dan y adalah "<<y<<endl;
17      cout<<"Nilai z yang telah diberikan adalah
          "<<z<<"\n\n";
18      system("PAUSE");
19      return EXIT_SUCCESS;
20  }
```

Hasil eksekusi:



Gambar 3.5 Hasil eksekusi program Lat 3.4 dalam bahasa C++

Catatan	Dalam bahasa C apabila tipe data diisi di luar jangkauan nilainya maka akan dihitung kembali dari jangkauan awal nilai sedangkan dalam bahasa C++ program tidak akan bisa dieksekusi karena dianggap nilai yang diberikan di luar jangkauan tipe data

Tipe Data Bilangan Pecahan

Tipe data ini adalah tipe yang digunakan untu merepresentasikan data-data bilangan yang mengandung angka di belakang koma. Sebagai contoh 56.86, 2810.86, dan sebagainya. Tipe data yang termasuk dalam kategori ini dapat dilihat pada tabel berikut:

Tabel 3.3 Tabel tipe data bilangan pecahan

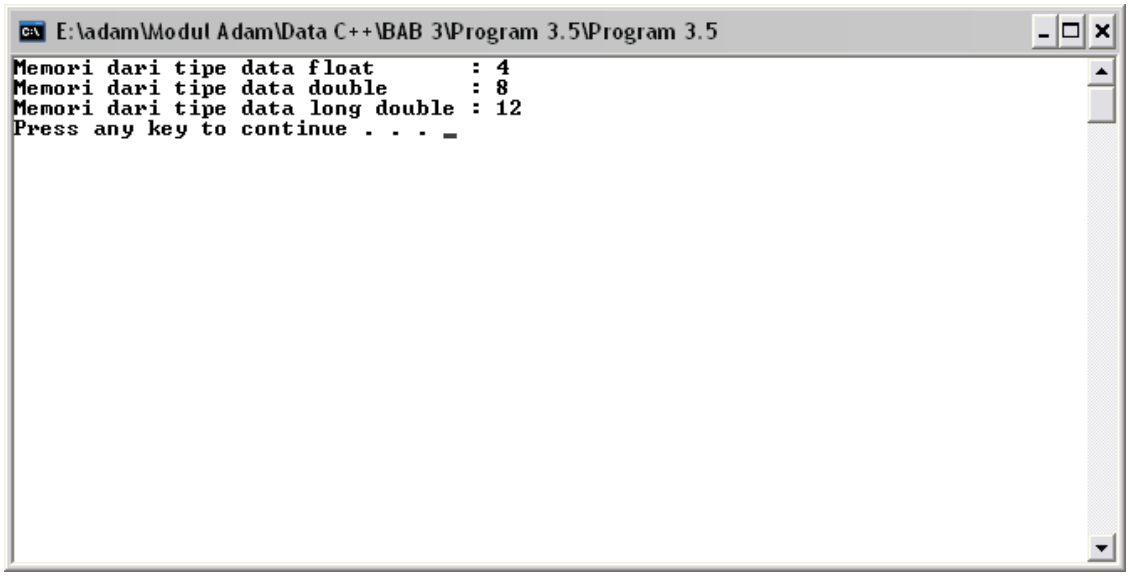
Tipe Data	Memori (Dalam Byte)	Format	Rentang	Digit Presisi
float	4	%f	1.2E-38 sampai 3.4E+38	6 digit
double	8	%f	2.3E-308 sampai 1.7E+308	15 digit
long double	12	%lf	3.4E-4932 sampai 1.1E+4932	19 digit

Untuk membuktikan memori yang digunakan per masing-masing tipe data, buatlah program dengan sintaks di bawah ini:

```
1  /*
2    Program 3.5
3    Nama File  : Lat-3.5.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12   printf("Memori dari tipe data float       :
           %i\n",sizeof(float));
13   printf("Memori dari tipe data double      :
           %i\n",sizeof(double));
14   printf("Memori dari tipe data long double :
           %i\n",sizeof(long double));
15   system("PAUSE");
16   return 0;
17 }
```

```
1   /*
2     Program 3.5
3     Nama File  : Lat-3.5.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
12  int main(int argc, char *argv[])
    {
13      cout<<"Memori dari tipe data float       :
           "<<sizeof(float)<<endl;
14      cout<<"Memori dari tipe data double      :
           "<<sizeof(double)<<endl;
15      cout<<"Memori dari tipe data long double :
           "<<sizeof(long double)<<endl;
17      system("PAUSE");
18      return EXIT_SUCCESS;
19  }
```

Hasil eksekusi:

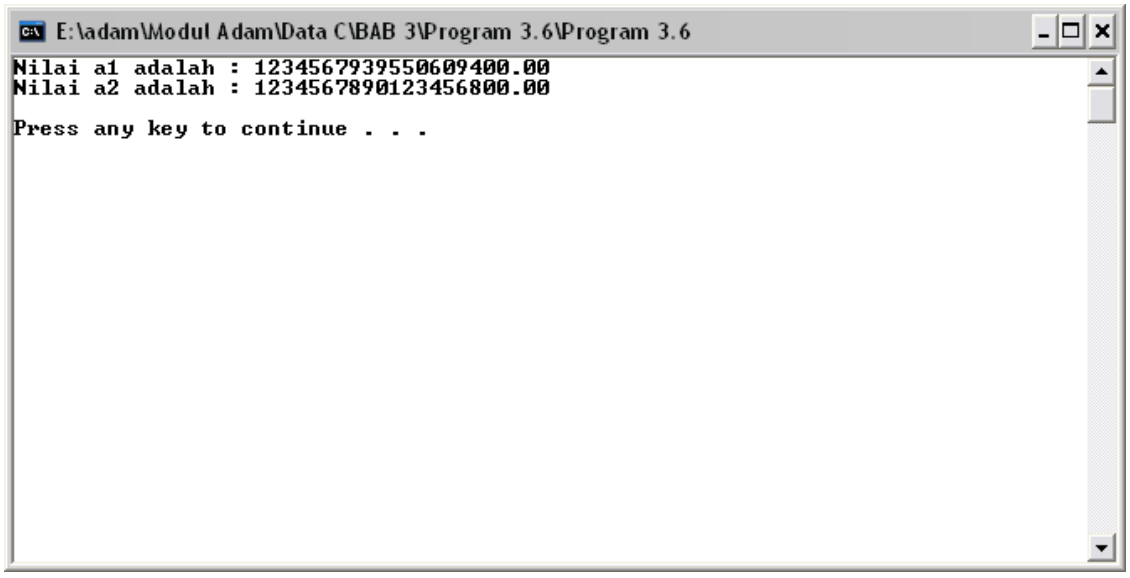


Gambar 3.6 Hasil eksekusi program Lat 3.5

Contoh program:

```
1   /*
2     Program 3.6
3     Nama File  : Lat-3.6.c
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <stdio.h>
8   #include <stdlib.h>
9
10  int main(int argc, char *argv[])
11  {
12      float a1=1234567890123456789;
13      double a2=1234567890123456789;
14      printf("Nilai a1 adalah : %.2f\n",a1);
15      printf("Nilai a2 adalah : %.2f\n\n",a2);
16      system("PAUSE");
17      return 0;
18  }
```

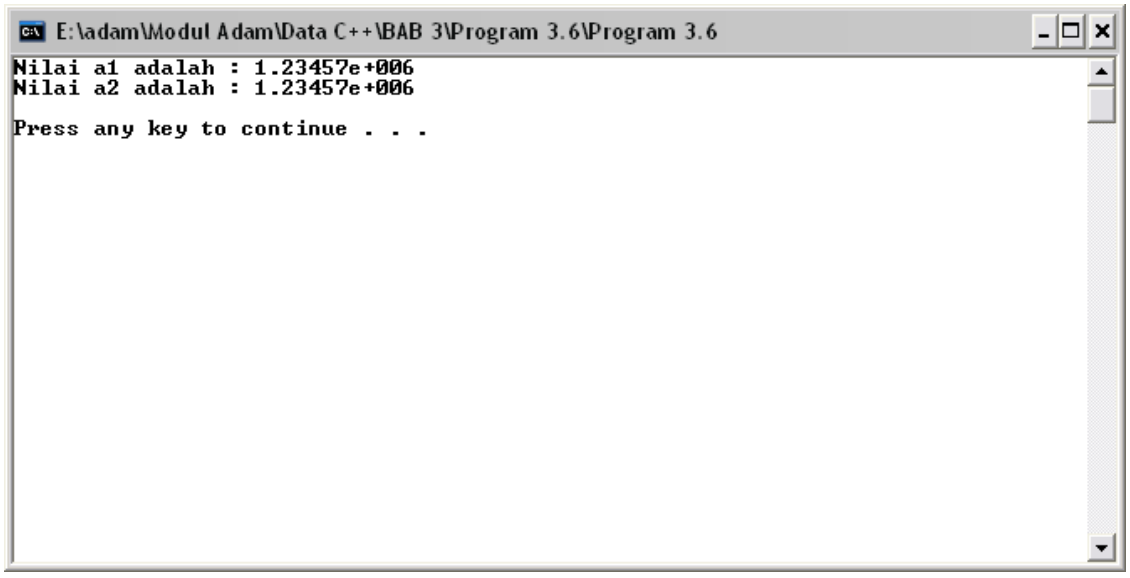
Hasil eksekusi:



Gambar 3.7 Hasil eksekusi program Lat 3.6 dalam bahasa C

```
1   /*
2     Program 3.6
3     Nama File  : Lat-3.6.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
12  int main(int argc, char *argv[])
13  {
14      float a1=1234567;
15      double a2=1234567;
16      cout<<"Nilai a1 adalah : "<<a1<<endl;
17      cout<<"Nilai a2 adalah : "<<a2<<"\n\n";
18      system("PAUSE");
19      return EXIT_SUCCESS;
20  }
```

Hasil eksekusi:



Gambar 3.8 Hasil eksekusi program Lat 3.6 dalam bahasa C++

Tipe Data String

Dalam pemrograman C, tidak ada perintah khusus untuk menampung data yang bertipe string. String akan dianggap sebagai kumpulan karakter yang diakhiri dengan karakter kosong. Oleh sebab itu string sering disebut juga sebagai sebuah array karakter atau sebuah pointer ke sebuah variabel char.

Cara pendeklarasiannya adalah sebagai berikut:

1. char nama[50];

   cara yang pertama ini kita memesan tempat sebanyak ukuran yang dideklarasikan (dalam kasus ini sebanyak 50 byte). Tapi kita hanya bisa mengisi variabel nama sebanyak 49 karakter. Hal ini dikarenakan tempat terakhir yaitu byte ke 50 akan diisi dengan karakter kosong (null atau \0). Jadi apabila ingin memesan memori untuk tipe data string, pesanlah dengen rumus sebagai berikut:

   Jumlah memori pesanan = Jumlah karakter maksimum+1

2. char *pekerjaan;

   cara yang kedua adalah kita menggunakan sebuah variabel pointer. Untuk cara yang kedua ini akan dijelaskan dalam bab pointer.

Contoh program:

```
1  /*
2    Program 3.7
3    Nama File  : Lat-3.7.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12     char nama[31];
13     char pekerjaan[21];
14     printf("Nama      : ");fflush(stdin);
15     scanf("%s",nama);
16     printf("Pekerjaan : ");fflush(stdin);
17     gets(pekerjaan);
18     printf("\nData yang telah dimasukan adalah : \n\n");
```

```
19      printf("Nama      : %s\nPekerjaan :
                %s\n\n",nama,pekerjaan);
20      system("PAUSE");
21      return 0;
22  }
```

Penjelasan:

1. fflush(stdin)

   perintah ini digunakan untuk memesan buffer (memori penyangga) untuk menampung data string dari keyboard secara temporer untuk nantinya dimasukkan ke dalam variabel yang ditentukan. Apabila perintah ini dihilangkan maka setelah pengguna memasukkan data string, program akan mengalami error.

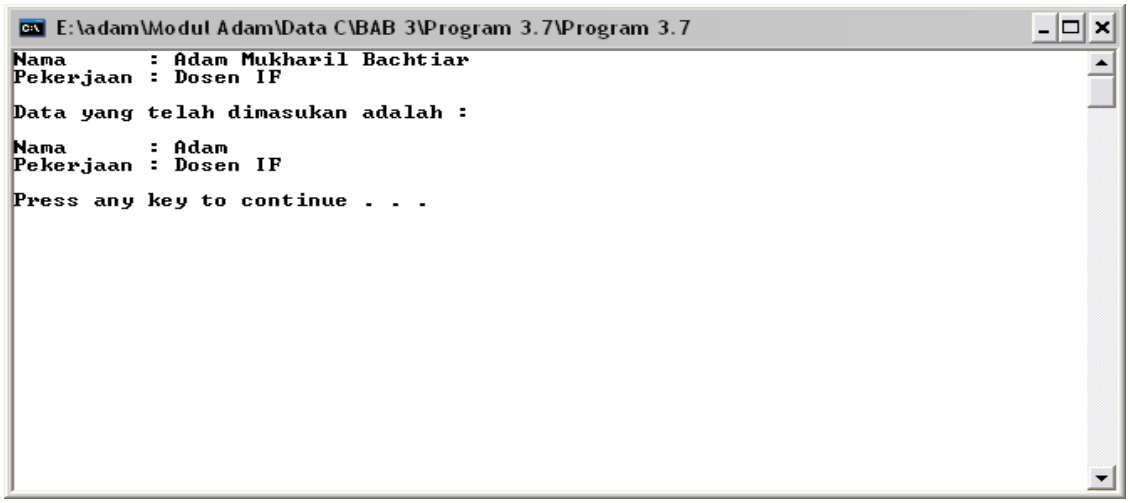
2. scanf(“&s”,nama);

   fungsi scanf digunakan untuk membaca masukkan dari keyboard. Kita harus menentukan format tipe data apa yang akan diinputkan. Untuk selanjutnya akan dibahas pada bab IV. Fungsi ini hanya bisa membaca string satu kata saja (tidak bisa mengandung spasi).

3. gets(pekerjaan)

   fungsi ini digunakan untuk membaca masukkan dari keyboard berupa data string baik yang mengandung spasi maupun yang tidak mengandung spasi.

Hasil eksekusi:



Gambar 3.9 Hasil eksekusi program Lat 3.7 dalam bahasa C

```
1  /*
2    Program 3.7
3    Nama File  : Lat-3.7.cpp
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
10 using namespace std;
11
12 int main(int argc, char *argv[])
13 {
14     char nama[31];
15     char pekerjaan[21];
16     cout<<"Nama       : ";cin.get(nama,30);
17     cout<<"Pekerjaan  : ";cin>>pekerjaan;
18     cout<<"\nData yang telah dimasukan adalah : \n\n";
19     cout<<"Nama       : "<<nama<<endl;
20     cout<<"Pekerjaan  : "<<pekerjaan<<"\n\n";
21     system("PAUSE");
22     return EXIT_SUCCESS;
23 }
```

Penjelasan:

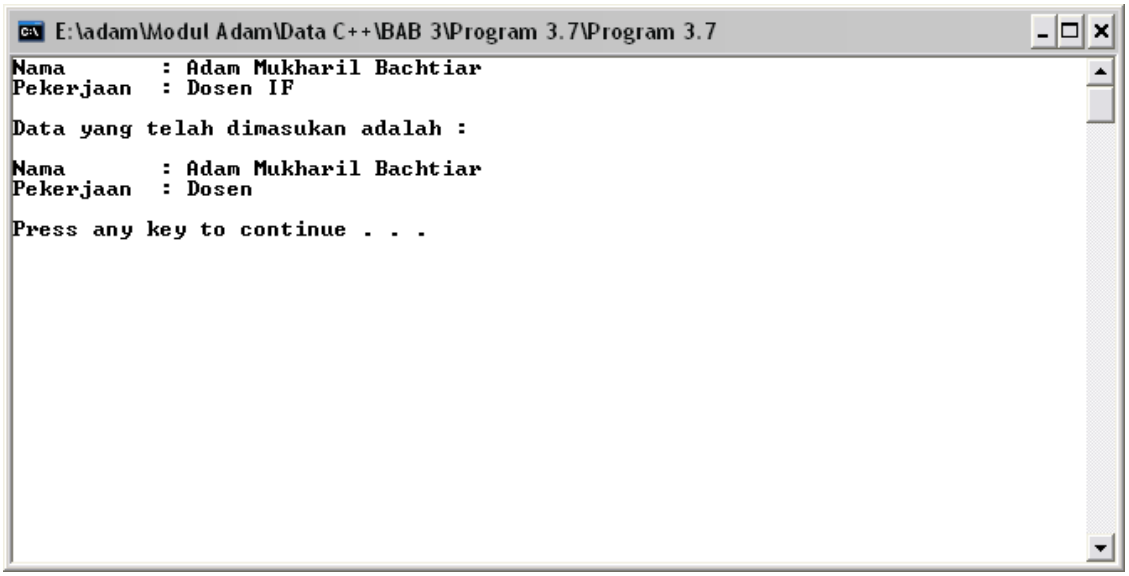
1. cin.get(nama,30);

   Fungsi ini digunakan untuk membaca inputan dari keyboard berupa string sebanyak karakter yang ditentukan (dalam kasus ini sebanyak 30 karakter) baik yang mengandung spasi maupun tidak mengandung spasi.

2. cin>>pekerjaan;

   Fungsi ini digunakan untuk membaca inputan dari keyboard (tidak hanya string) tetapi tidak mendukung spasi.

Hasil eksekusi:



Gambar 3.10 Hasil eksekusi program Lat 3.7 dalam bahasa C++

Identifier (Pengenal)

Data pada bahasa C dan bahasa C++ tersusun dari:

1. Variabel
2. Konstanta.

Variabel merupakan komponen penting pada pemrograman. Variabel digunakan untuk menampung suatu nilai, dan nilai yang ada padanya dapat diubah selama eksekusi program berlangsung. Variabel dan konstanta lebih dikenal sebagai identifier.

Dalam menentukan identifier ada beberapa hal yang harus diperhatikan, antara lain:

1. Bahasa C dan C++ bersifat case sensitive, jadi kompiler akan membedakan antara variabel yang ditulis dengan huruf besar dengan variabel yang ditulis dengan huruf kecil. Contoh:
   a. **Nama** berbeda dengan **nama**
   b. **PEKERJAAN** berbeda dengan **pekerjaan**.
2. Identifier tidak boleh berupa angka atau diawali dengan karakter yang berupa angka. Contoh:
   a. int 500 → salah.
   b. int 2a → salah.

c. int a2 → benar.

3. Identifier tidak boleh mengandung spasi. Contoh:
   a. int bilangan pertama → salah.
   b. int _bilanganpertama → benar.
   c. int bilangan_pertama → benar.
4. Identifier tidak boleh mengandung karakter-karakter simbol (#, ?, !, @, $, dll). Contoh:
   a. int !x → salah.
   b. int y? → salah.
   c. int z#w → salah.
5. Identifier tidak boleh menggunakan kata kunci di dalam bahasa C maupun C++. Adapun daftar kata kunci dalam bahasa C dan C++ antara lain:

• asm	• float	• short
• auto	• for	• signed
• break	• friend	• sizeof
• case	• goto	• static
• char	• if	• struct
• class	• inline	• switch
• const	• int	• template
• continue	• long	• this
• default	• new	• typedef
• delete	• operator	• union
• do	• private	• unsigned
• double	• protected	• virtual
• else	• public	• void
• enum	• register	• volatile
• extern	• return	• while

6. Nama identifier usahakan sederhana dan mudah diingat.
7. Nama identifier tidak boleh sama.

Konstanta

Konstanta adalah jenis identifier yang bersifat konstan atau tetap. Nilai dari konstanta di dalam program tidak dapat diubah. Konstanta berguna untuk menentukan nilai tetapan, sebagai contoh pi (), kecepatan cahaya, dan lain-lain.

Ada dua buah cara untuk membuat sebuah konstanta, yaitu:

1. Menggunakan #define

   Cara penggunaanya adalah sebagai berikut:

   ```
   #define nama_konstanta nilai_konstanta;
   atau
   #define nama_konstanta nilai_konstanta
   ```

   Sebagai contoh:

   a. #define phi 3.14
   b. #define terbesar 100;
   c. #define biru BLUE;

2. Menggunakan kata kunci const

   Cara penggunaannya adalah sebagai berikut:

   ```
   const  tipe_data nama_kostanta = nilai_tetapan;
   ```

   Sebagai contoh:

   a. const float pi = 3.14;
   b. const int bil_pertama = 0;
   c. const char universitas[10] = “unikom”;

Variabel

Variabel sangat berbeda dengan konstanta. Perbedaannya terletak pada nilai. Variabel mempunyai nilai yang dinamis. Arti kata dinamis di sini bahwa nilai variabel tersebut dapat diubah sesuai kebutuhan dalam program. Cara pendeklarasiannya adalah sebagai berikut:

```
tipe_data nama_variabel;
atau
tipe data nama_variabel1, nama_variabel2, nama_variabel3;
```

Sebagai contoh:

1. int x;
2. int x,y,z;
3. int x=5, y=6, z=86;

Variabel Global

Variabel global dapat dikenali oleh semua lingkungan dalam program yang dibuat. Sebagai contoh ada tiga fungsi yang terdapat di dalam program maka ketiga fungsi tersebut dapat mengenali variabel global tanpa harus mendeklarasikan ulang di dalam fungsi masing-masing.

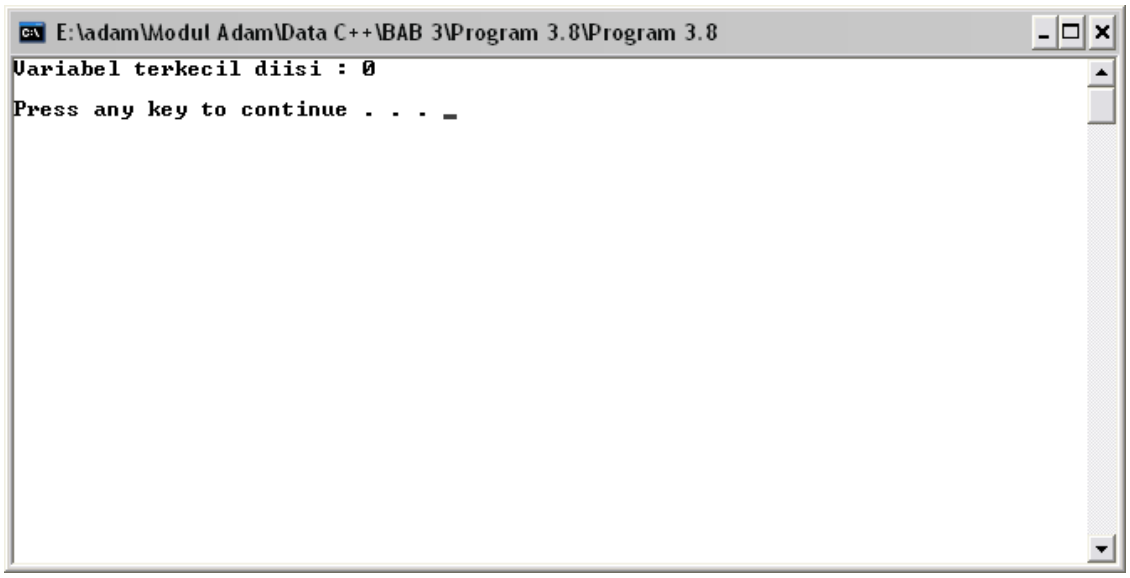
Variabel global dideklarasikan di luar seluruh fungsi yang ada. Apabila fungsi yang ada hanya fungsi main() maka pendeklarasian variabel harus di luar dari fungsi main(). Sebagai contoh buat program dengan sintaks di bawah ini:

```
1  /*
2    Program 3.8
3    Nama File  : Lat-3.8.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int terkecil;
11
12 int main(int argc, char *argv[])
13 {
14   terkecil=0;
15   printf("Variabel terkecil diisi : %i\n\n",terkecil);
16   system("PAUSE");
17   return 0;
18 }
```

```
1  /*
2    Program 3.8
3    Nama File  : Lat-3.8.cpp
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
10 using namespace std;
11
12 int terkecil;
```

```
13
14  int main(int argc, char *argv[])
15  {
16      terkecil=0;
17      cout<<"Variabel terkecil diisi : "<<terkecil
            <<"\n\n";
18      system("PAUSE");
19      return EXIT_SUCCESS;
20  }
```

Hasil eksekusi:



Gambar 3.11 Hasil eksekusi program Lat 3.8

Variabel Lokal

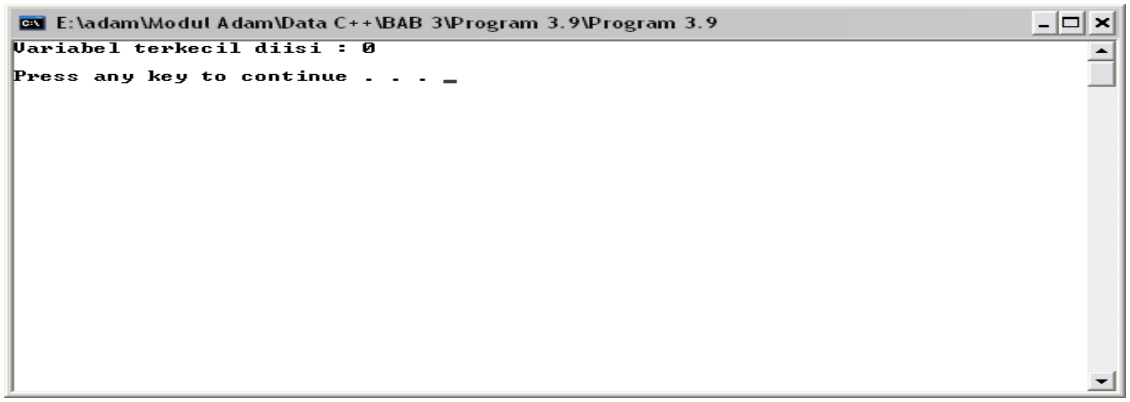
Varianel lokal hanya dikenali di dalam fungsi yang mendeklarasikannya saja. Artinya tidak dikenal oleh lingkungan luar dari fungsi tersebut. Untuk lebih jelasnya buat program dengan sintaks di bawah ini:

```
1   /*
2     Program 3.9
3     Nama File  : Lat-3.9.c
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <stdio.h>
8   #include <stdlib.h>
```

```
9
10  int main(int argc, char *argv[])
11  {
12    int terkecil;
13    terkecil=0;
14    printf("Variabel terkecil diisi : %i\n\n",terkecil);
15    system("PAUSE");
16    return 0;
17  }
```

```
1   /*
2     Program 3.9
3     Nama File  : Lat-3.9.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
12  int main(int argc, char *argv[])
13  {
14      int terkecil;
15      terkecil=0;
16      cout<<"Variabel terkecil diisi : "<<terkecil
            <<"\n\n";
17      system("PAUSE");
18      return EXIT_SUCCESS;
19  }
```

Hasil eksekusi:



Gambar 3.12 Hasil eksekusi program Lat 3.9

Variabel Statis

Variabel statis adalah variabel yang menempati ruang memori komputer secara permanen, artinya nilai terakhir dari variabel ini akan terus disimpan sampai eksekusi program selesai. Berbeda dengan variabel biasa yang datanya akan tidak akan dipertahankan. Cara pendeklarasiannya adalah sebagai berikut:

```
static tipe_data nama_variabel;
```

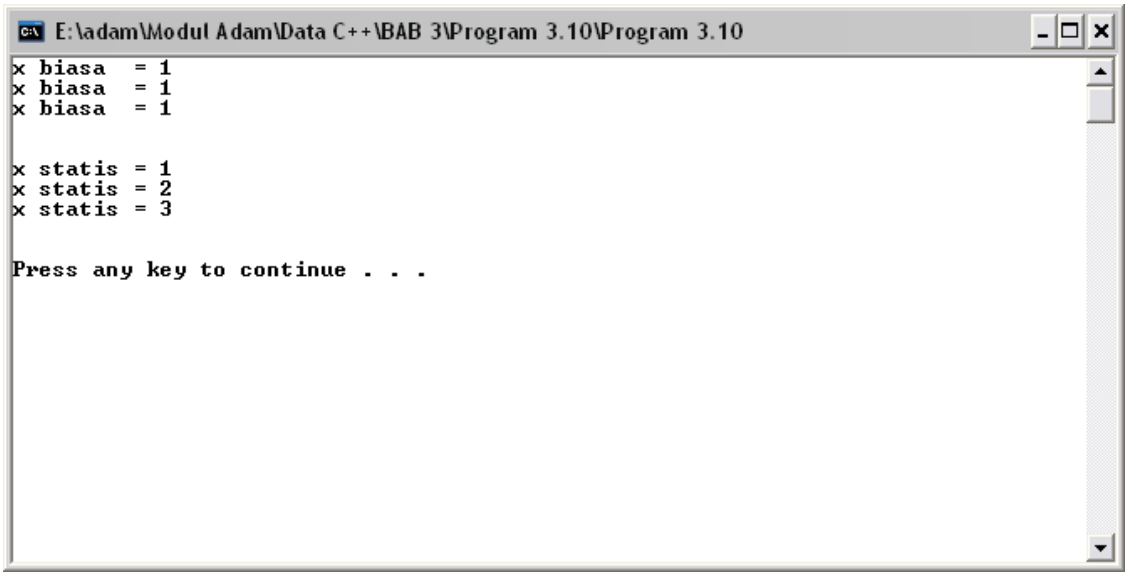
Sebagai contoh buat program berikut:

```
1  /*
2    Program 3.10
3    Nama File  : Lat-3.10.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 void contoh_biasa()
11 {
12      int x=0;
13      x=x+1;
14      printf("x biasa  = %i\n",x);
15 }
16
17 void contoh_statis()
18 {
19      static int x=0;
20      x=x+1;
21      printf("x statis = %i\n",x);
22 }
23
24 int main(int argc, char *argv[])
25 {
26   contoh_biasa();
27   contoh_biasa();
28   contoh_biasa();
29   printf("\n\n");
30   contoh_statis();
31   contoh_statis();
32   contoh_statis();
33   printf("\n\n");
```

```
34    system("PAUSE");
35    return 0;
36  }
```

```
1   /*
2     Program 3.10
3     Nama File  : Lat-3.10.cpp
4     Programmer : Adam Mukharil Bachtiar, S.Kom.
5   */
6
7   #include <cstdlib>
8   #include <iostream>
9
10  using namespace std;
11
12  void contoh_biasa()
13  {
14       int x=0;
15       x=x+1;
16       cout<<"x biasa  = "<<x<<endl;
17  }
18
19  void contoh_statis()
20  {
21       static int x=0;
22       x=x+1;
23       cout<<"x statis = "<<x<<endl;
24  }
25
26  int main(int argc, char *argv[])
27  {
28      contoh_biasa();
29      contoh_biasa();
30      contoh_biasa();
31      cout<<"\n\n";
32      contoh_statis();
33      contoh_statis();
34      contoh_statis();
35      cout<<"\n\n";
36      system("PAUSE");
37      return EXIT_SUCCESS;
38  }
```

Hasil eksekusi:



Gambar 3.13 Hasil eksekusi program Lat 3.10

Operator-Operator Perhitungan

Setelah mempelajari tipe data dan variabel maka variabel-variabel tersebut nantinya akan dikenai operasi perhitungan. Untuk melakukan operasi perhitungan diperlukan adanya operator-operator perhitungan. Adapun operator-operator perhitungan yang sering digunakan antara lain:

Tabel 3.4 Tabel operator-operator perhitungan

Operator	Contoh	Arti
=	a=b	Variabel a diisi dengan nilai dari variabel b
+	c=a+b	Variabel c diisi dengan isi variabel a ditambah isi variabel b
-	c=a-b	Variabel c diisi dengan isi variabel a dikurangi isi variabel b
*	c=a*b	Variabel c diisi dengan isi variabel a dikali isi variabel b
/	c=a/b	Variabel c diisi dengan isi variabel a dibagi isi variabel b
%	c=a%b	Variabel c diisi dengan sisa pembagian variabel a dibagi isi variabel b
++	a++, ++a	Isi variable a ditambah 1. Perintah ini sama dengan a=a+1 atau a+=1
--	b--, --b	Isi variable b dikurang. Perintah ini sama dengan b=b-1 atau b-=1

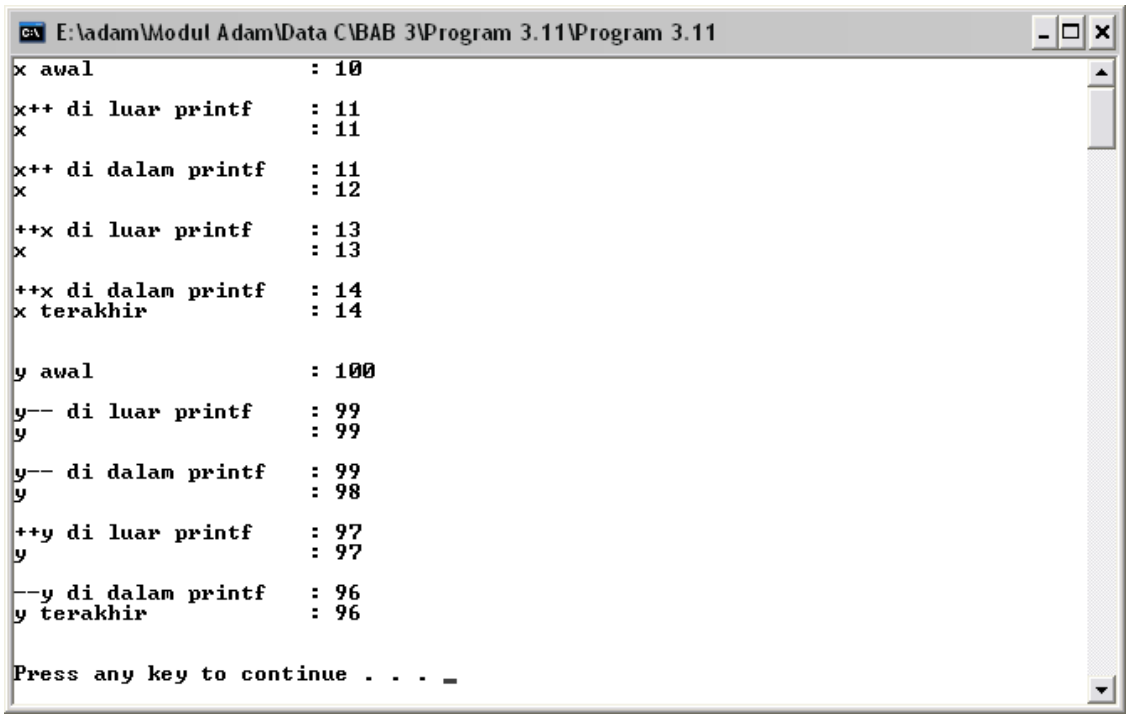
+ =	c+=a	Variable c ditambah dengan isi variable a. Sama dengan c=c+a
/ =	c/=a	Variable c dibagi dengan isi variable a. Sama dengan c=c/a
-=	c-=a	Variable c dikurangi dengan isi variable a. Sama dengan c=c -a
*** =**	c*=a	Variable c dikali dengan isi variable a. Sama dengan c=c*a
% =	c%=a	Variable c diisi dari sisa pembagian c dibagi isi variable a. Sama dengan c=c%a

Yang harus diperhatikan adalah penggunaan operator ++ dan --. Peletakan operator ini berpengaruh terhadap hasil perhitungan. Untuk lebih jelasnya buat program dengan sintaks di bawah ini:

```
1  /*
2    Program 3.11
3    Nama File  : Lat-3.11.c
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main(int argc, char *argv[])
11 {
12   int x=10;
13   int y=100;
14   printf("x awal                : %i\n\n",x);
15   x++;
16   printf("x++ di luar printf    : %i\n",x);
17   printf("x                     : %i\n\n",x);
18   printf("x++ di dalam printf   : %i\n",x++);
19   printf("x                     : %i\n\n",x);
20   ++x;
21   printf("++x di luar printf    : %i\n",x);
22   printf("x                     : %i\n\n",x);
23   printf("++x di dalam printf   : %i\n",++x);
24   printf("x terakhir            : %i\n\n\n",x);
25   printf("y awal                : %i\n\n",y);
26   y--;
27   printf("y-- di luar printf    : %i\n",y);
28   printf("y                     : %i\n\n",x);
29   printf("y-- di dalam printf   : %i\n",y--);
```

```
30    printf("y                       : %i\n\n",y);
31    --y;
32    printf("++y di luar printf      : %i\n",y);
33    printf("y                       : %i\n\n",y);
34    printf("--y di dalam printf     : %i\n",--y);
35    printf("y terakhir              : %i\n\n\n",y);
36    system("PAUSE");
37    return 0;
38  }
```

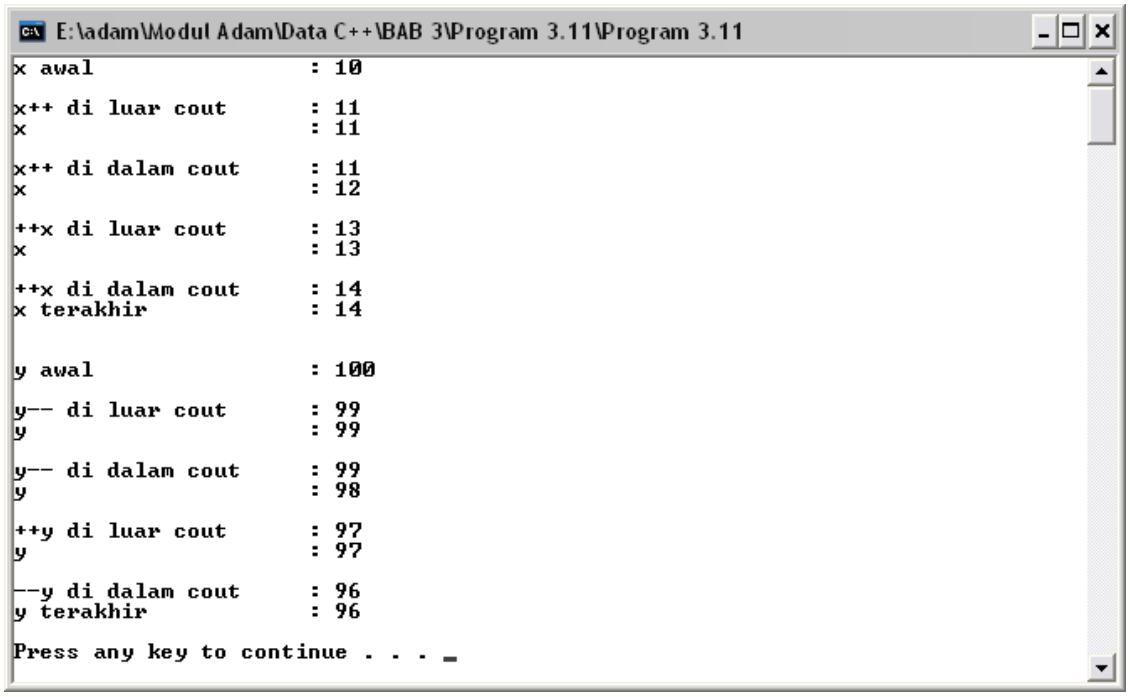
Hasil eksekusi:



Gambar 3.14 Hasil eksekusi program Lat 3.11 dalam bahasa C

```
1  /*
2    Program 3.11
3    Nama File  : Lat-3.11.cpp
4    Programmer : Adam Mukharil Bachtiar, S.Kom.
5  */
6
7  #include <cstdlib>
8  #include <iostream>
9
10 using namespace std;
11
12 int main(int argc, char *argv[])
13 {
14     int x=10;
15     int y=100;
16     cout<<"x awal                : "<<x<<"\n\n";
17     x++;
18     cout<<"x++ di luar cout      : "<<x<<"\n";
19     cout<<"x                     : "<<x<<"\n\n";
20     cout<<"x++ di dalam cout     : "<<x++<<"\n";
21     cout<<"x                     : "<<x<<"\n\n";
22     ++x;
23     cout<<"++x di luar cout      : "<<x<<"\n";
24     cout<<"x                     : "<<x<<"\n\n";
25     cout<<"++x di dalam cout     : "<<++x<<"\n";
26     cout<<"x terakhir            : "<<x<<"\n\n\n";
27     cout<<"y awal                : "<<y<<"\n\n";
28     y--;
29     cout<<"y-- di luar cout      : "<<y<<"\n";
30     cout<<"y                     : "<<y<<"\n\n";
31     cout<<"y-- di dalam cout     : "<<y--<<"\n";
32     cout<<"y                     : "<<y<<"\n\n";
33     --y;
34     cout<<"++y di luar cout      : "<<y<<"\n";
35     cout<<"y                     : "<<y<<"\n\n";
36     cout<<"--y di dalam cout     : "<<--y<<"\n";
37     cout<<"y terakhir            : "<<y<<"\n\n";
38     system("PAUSE");
39     return EXIT_SUCCESS;
40 }
```

Hasil eksekusi:



Gambar 3.15 Hasil eksekusi program Lat 3.11 dalam bahasa C++