ALGORITMA & PEMROGRAMAN #2

TIPE DATA, KONSTANTA,VARIABEL
&
EKSPRESI

Sufajar Butsianto, M.Kom

Rev.00

BAHASA MESIN

- **Level terendah**
- **Isi:**
  - *kode-kode mesin yg hanya dapat di interpretasikan langsung oleh mesin komputer*
- **Berupa kode numerik 0 dan1**

- **Microcode :**
  - Sekumpulan instruksi dalam bahasa mesin

- **(+): Eksekusi cepat**
- **(-): Sulit dipelajari manusia**

BAHASA ASSEMBLY

- **Bahasa simbol dari bahasa mesin**
- **Contoh: ADD, MUL, SUB, dll**
- **Macro instruksi :**
  - *sekumpulan kode dalam bahasa assembly*
- **(+): Eksekusi cepat, masih dapat dipelajari daripada bahasa mesin, file kecil**
- **(-): Tetap sulit dipelajari, program sangat panjang**

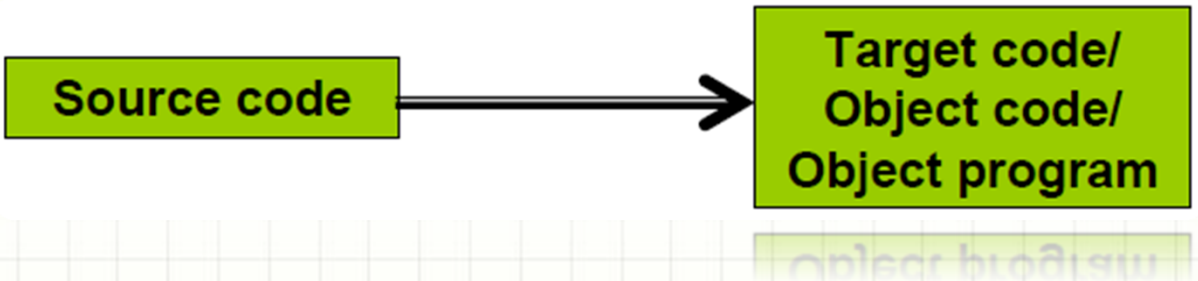
BAHASA TINGKAT TINGGI

- The 3 rd Generation Programming Language
- Lebih dekat dengan bahasa manusia
- Memberi banyak fasilitas kemudahan dalam pembuatan program, mis.: variabel, tipedata, konstanta, strukturkontrol, loop, fungsi, prosedur, dll.
- Contoh: Pascal, Basic, C++, Java
- (+): Mudah dipelajari, mendekati permasalahan yang akan dipecahkan, kode program pendek
- (-): Eksekusi lambat

SPECIFIC PROBLEM ORIENTED

- **The 4 th Generation Programming Language**
- **Digunakan langsung untuk memecahkan suatu masalah tertentu**
- **SQL untuk database, Visual Basic, Delphi, Visual Studio, Visual C++**

TRANSLATOR



- **Source Code:**
  - *Ditulis dengan bahasa pemrograman tertentu*
- **Object Code:**
  - *bisa bermacam-macam, tergantung pada translator-nya*

MACAM TRANSLATOR

- Assembler
- Source code adalah bahasa assembly
- Object code adalah bahasa mesin

INTERPRETER(2)

- **Program tidak harus dianalisis seluruhnya dulu tapi bersamaan dengan jalannya program**
- **(+)**
  - *Mudah bagi user*
  - *Debugging cepat*
- **(-)**
  - *eksekusi program lambat*
  - *tidak langsung menjadi program*

MACAM TRANSLATOR

KOMPILER

- **Input**
  - *source code: bahasa Pascal, C, C++*
- **Output**
  - *object code: bahas aassembly atau*

MACAM TRANSLATOR

KOMPILER(2)

❑ Compile Time

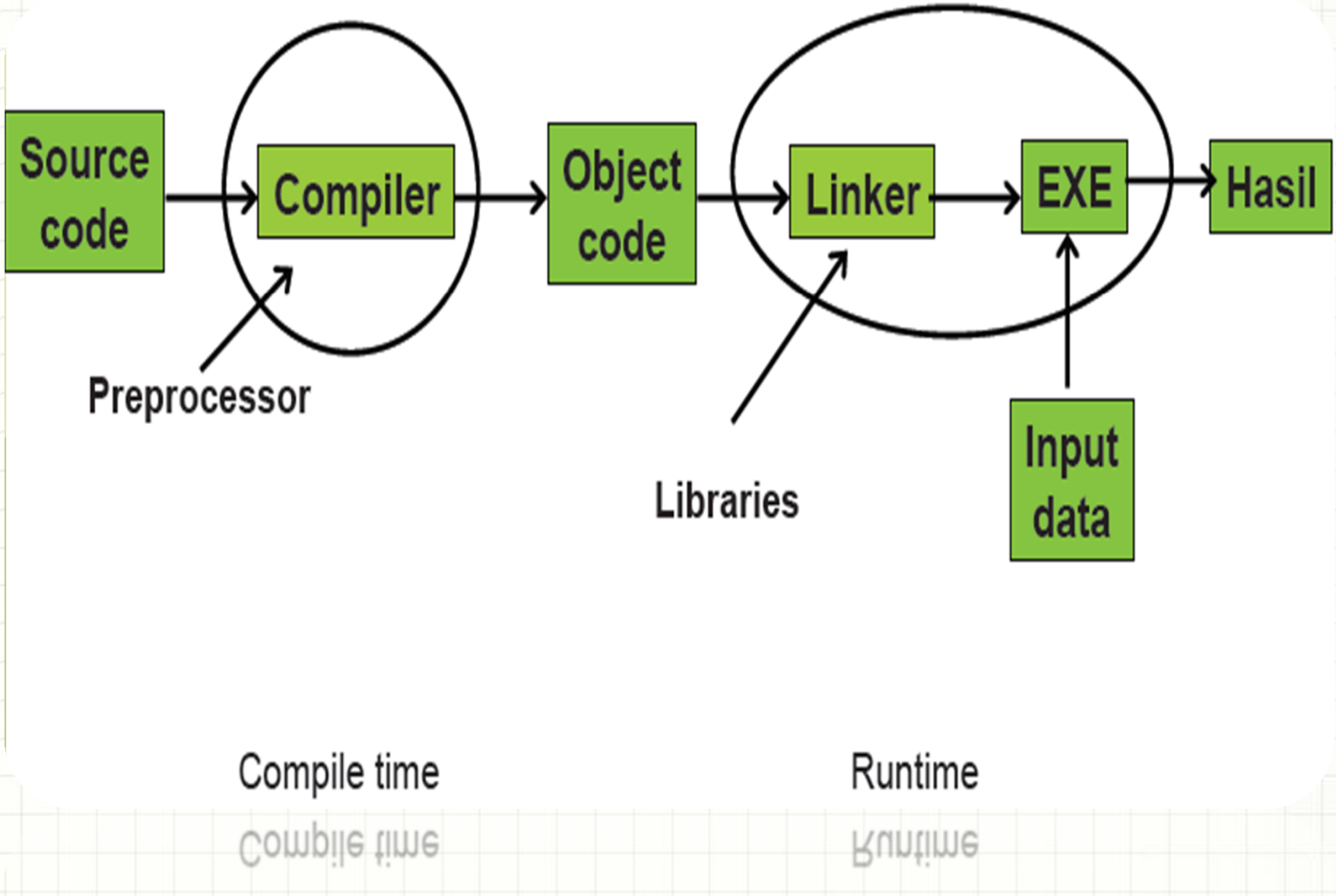
- ➢ *saat pengubahan source code menjadi object code*

❑ Runtime

- ➢ *saat eksekusi object code, (dan menerima input dari user)*

TRANSLATOR

KOMPILER(2)



Bahasa C

- Bahasa pemrograman tingkat menengah
- 1972:
  - Dirancang oleh **Dennis M Ritchie** di **Bell Laboratories**
- 1978:
  - Dennis dan Brian W. Kernighan mempublikasikan bahasa C melalui “The C Programming Language”
- 1989:
  - Bahasa C distandarisasi ANSI

Contoh Program

```
#include <stdio.h>

void main()
{
   printf("Halo! Selamat Belajar C");
}
```

Bahasa C

- Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian (*subroutine/ module*).
- Fungsi-fungsi selain fungsi utama disebut *subroutine/ module* dan ditulis setelah fungsi utama (**main**) atau diletakkan pada file pustaka (*library*).
- Jika fungsi-fungsi diletakkan pada file pustaka dan akan dipakai disuatu program, maka nama file *header*nya harus dilibatkan dalam program menggunakan *preprocessor directive* **#include**

Bahasa C

- Struktur Program C adalah:
  - Suatu program C minimal harus memiliki function **main()**, tanpa function itu maka program C **tidak dapat dieksekusi** tapi **bisa dikompilasi**.

```
<preprosessor directive>

void main(){

      <statement>;

      <statement>;

      <statement>;

}
```

```
<preprosessor directive>
int main(){
      <statement>;
      <statement>;
      <statement>;
      return 0;
}
```

Statement & Preprosesor Directive

- Statement adalah suatu baris instruksi/perintah tertentu. Statement menyebabkan suatu tindakan akan dilakukan oleh komputer.
- Preprosessor Directive adalah bagian yang berisi pengikutsertaan file atau berkas-berkas fungsi maupun pendefinisian konstanta atau fungsi makro tertentu.

Contoh suatu program C (2)

```
#include <stdio.h>
int main()
{
    int a,b,c;
    printf("Enter the first value:");
    scanf("%d",&a);
    printf("Enter the second value:");
    scanf("%d",&b);
    c = a + b;
    printf("%d + %d = %d\n",a,b,c);
    return 0
}
```

Keterangan

- Deklarasi variabel menyebabkan komputer menyediakan tempat yang diberi nama (*identifier*) **a**, **b** dan **c** dengan ukuran integer (2 byte = 16 bit).
- **printf** akan membuat komputer mengirim teks yang berada dalam fungsi tersebut ke layar monitor, sedangkan **scanf** membuat komputer menanti masukan dari pemakai melalui *keyboard*.

Keterangan (2)

- Pada program ini akan dikerjakan proses aritmatika, yaitu proses memberi nilai (*assignment* yang dipakai tanda "=") variabel "c" dengan nilai yang ada dalam variabel "a" ditambah nilai yang ada dalam variabel "b"
- Yang terakhir adalah proses mencetak ke layar monitor dengan format yang sesuai

Statement

Contoh Statement

Instruksi/ Statement	*Tindakan*
`A = b * c ;`	Menghitung
`printf("Antonius Rachmat C");`	Menampilkan literal string
`scanf("%f",&Celcius);`	Menerima input data
`if(N<0) printf("negatif");`	Mengendalikan proses

Jenis Statement

- Macam:
  1. Statement kosong
  2. Statement ungkapan
  3. Statement kendali
  4. Statement jamak

Statement Kosong

- Empty statement = null statement
- Statement yang hanya terdiri dari pengakhir titik koma (;) saja
- Tidak ada tindakan yang akan dilakukan
- Contoh:
  - Memberi jarak waktu/delay

```
For (J=0; J<50000; J++);
```

Statement Ungkapan

- Expression statement
- Statement yang dibentuk dari suatu ungkapan
- Diakhiri dengan titik koma (;)
- Contoh:

```
scanf("%f",&Panjang);
scanf("%f",&Lebar);
Luas=Panjang*Lebar;
X=Y;
Y=Y+1;
```

Statement Kendali

- Control statement
- Statement yang digunakan untuk mengendalikan proses dari program, yaitu:
  - Penyeleksian kondisi (percabangan):
    - If, case dan switch
  - Lompatan (perulangan)
    - for, while, do-while, goto, break dan continue
- Contoh:

```
if (N<0) printf("Nilai N negatif");
```

Statement Jamak

- Compound statement = block statement
- Statement yang terdiri dari gabungan beberapa statement tunggal yang ditulis pada posisi di antara tanda kurung kurawal (**"{"** dan **"}"**)
- Contoh:

```
{ scanf("%f",&Panjang);
  scanf("%f",&Lebar);
  Luas=Panjang*Lebar;
  Printf("Luas = %f",Luas);
}
```

Struktur Program C (3)

- Selain function *main()* dapat ditambahkan function lain
- Jika function akan diletakkan di sembarang tempat dari function *main()*, maka function tersebut harus dideklarasikan terlebih dahulu sebelum function *main()*

```c
#include <stdio.h>
int jumlahkan(int a, int b);

void main()
{  printf("Hasil 5 + 3 adalah %d", jumlahkan(5,3));
}

int jumlahkan(int a, int b)
{  return a+b;
}
```

```c
#include <stdio.h>

int jumlahkan(int a, int b)
{   return a+b;
}

void main()
{  printf("Hasil 5 + 3 adalah %d", jumlahkan(5,3));
}
```

Identifier

- Identifier:
  - suatu tempat untuk menyimpan nilai
  - Diberi nama unik dan bisa memiliki tipe data
  - Dibagi menjadi 2:
    1. Konstanta
    2. Variabel
  - Dapat juga merupakan nama suatu elemen dalam program, mis.
    - Nama function
    - Nama prosedur
    - Nama tipe data, dll

Jenis Identifier

1. **Konstanta**
   - Identifier yang nilainya tetap selama program berjalan (dieksekusi)
   - Cara untuk mengubahnya hanya melalui *source code* saja

2. **Variabel**
   - Identifier yang nilainya dapat berubah atau diubah selama program berjalan (dieksekusi)
   - Pengubah: *user* atau proses

Standard Identifier

- Standard Identifier adalah identifier-identifier yang biasanya berupa fungsi-fungsi tertentu yang telah diberi makna tertentu oleh compiler bahasa C, tetapi tidak bersifat reserved sehingga masih bisa dipakai kembali oleh pemrogram.

- Contoh standard identifier :

```
#include <stdio.h>
#include <conio.h>
void main(){
    clrscr();
    printf("hallo bahasa C");
}
```

ATURAN PENULISAN IDENTIFIER

- Tidak boleh sama dengan nama keyword reserved, function, dan harus unik.
- Maksimum 32 karakter. Bila lebih, maka karakter selebihnya tidak akan diperhatikan oleh komputer.
- Case sensitive : membedakan huruf besar dan kecil
- Karakter pertama harus huruf atau underscore (_), selebihnya boleh angka.
- Tidak boleh mengandung spasi / blank

Keywords

- Adalah identifier yang telah didefinisikan oleh bahasa C
- Sifat:
  - Memiliki arti dan pemakaian tertentu
  - Reserved
  - Ditulis dalam huruf kecil
- Menurut standar ANSI: 32 keywords

Keywords (2)

auto	double	int	switch
break	else	long	typedef
case	enum	register	union
char	extern	return	unsigned
const	float	short	void
continue	for	signed	volatile
default	goto	sizeof	while
do	if	static	struct

Tipe Data

Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
short int	16 bits	-32,768 to 32,767
unsigned int	32 bits	0 to 4,294,967,295
int	32 bits	-2,147,483,648 to 2,147,483,648
unsigned long	32 bits	0 to 4,294,967,295
enum	16 bits	-2,147,483,648 to 2,147,483,648
long	32 bits	-2,147,483,648 to 2,147,483,648

Tipe Data (2)

Type	Length	Range
float	32 bits	3.4 x 10-38 to 3.4 x 10+38
double	64 bits	1.7 x 10-308 to 1.7 x 10+308
long double	80 bits	3.4 x 10-4932 to 3.4 x 10+4932
near (pointer)	32 bits	Not applicable
far (pointer)	32 bits	Not applicable

- Cara untuk mengetahui ukuran sebuah tipe data di C: `sizeoff(<tipe_data>)`

Bahasa C - Int

- Bilangan Bulat
- Rangenya : -32768 sampai 32767 (16 bit)
- Deklarasinya :
  - int a;
- Untuk memberi nilai :
  - a = 1000;
- Contoh operasi :
  - a = a + 1000; //berapa nilai a sekarang ?

Bahasa C – Float & Double

- Bilangan real (pecahan, floating point)
- Float
  - Nilainya antara 1 E -36 sampai 1 E 36
  - Presisi 7 digit
  - 32 bit (4 byte)
- Double
  - Nilainya antara 1 E -303 sampai 1 E 303
  - Presisi 13 digit
  - 64 bit (8 byte)

Bahasa C – Float & Double

- Deklarasinya :
  - float dataku;
  - double luaskubus;
- Untuk memberi nilai :
  - dataku = 3.245;
  - luaskubus = 4.56789665;

TERIMA KASIH