



**C LANGUAGE**

In the computer world, the further a programming language is from the computer architecture, the higher the language's level. You can imagine that the lowest-level languages are machine languages that computers understand directly. The high-level programming languages, on the other hand, are closer to our human languages.

# Bahasa Tingkat Tinggi

- Ada
- Modula-2
- Pascal
- Cobol
- Fortran
- Basic

## Bahasa Tingkat Menengah

- Java
- C++
- C
- Forth

## Bahasa Tingkat Rendah

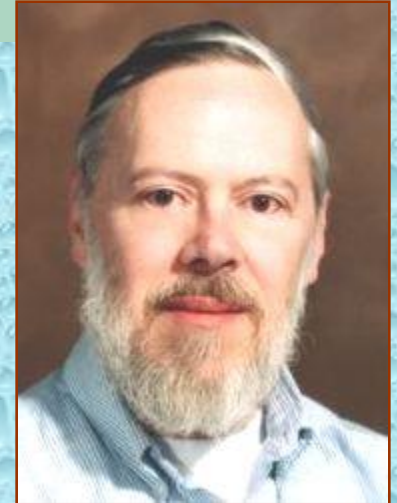
- Macro Assembler
- Assembler

Bahasa C adalah merupakan bahasa pemrograman tingkat menengah yaitu diantara bahasa tingkat rendah dan tingkat tinggi yang biasa disebut dengan **Bahasa Tingkat Tinggi dengan Perintah Assembly**



# SEJARAH BAHASA C

- Bahasa C dirancang oleh Dennis Mc Allistair Ritchie, AT&T Bell laboratories, 1972
- Dirancang untuk mengembangkan piranti lunak sistem
- Kernel sistem operasi UNIX ditulis kembali dengan bahasa C, 1973
- Publikasi melalui buku The C Programming Language, Brian W. Kernighan dan Dennis M. Ritchie, 1978



C have the following advantages:

- Readability: Programs are easy to read.
- Maintainability: Programs are easy to maintain.
- Portability: Programs are easy to port across different computer platforms.

Each high-level language needs a compiler or an interpreter to translate instructions written in the high-level programming language into a machine language that a computer can understand and execute. Different machines may need different compilers or interpreters for the same programming language.



A CPU has millions of transistors that make use of electronic switches.

The electronic switches have only two states: off and on. (Symbolically, 0 and 1 are used to represent the two states.) Therefore, a computer can only understand instructions consisting of series of 0s and 1s. In other words, machine-readable instructions have to be in binary format.

However, a computer program written in a high-level language, such as C, Java, or Perl, is just a text file, consisting of English-like characters and words. We have to use some special programs, called compilers or interpreters, to translate such a program into a machine-readable code. That is, the text format of all instructions written in a high-level language has to be converted into the binary format. The code obtained after the translation is called binary code. Prior to the translation, a program in text format is called source code.

# Tahapan penulisan program C

1. Menulis Naskah Program.
2. Mengkompilasi Program (***Compile***)
3. Melakukan Pengujian Program (***Debugging***)
4. Mengaitkan Object dan Library ke Program (***Linking***)
5. Menjalankan Program (***Running***)

# TIPIKAL PROGRAM C

```
/* potongan harga untuk 3 kaleng */
# include <stdio.h>
int main () {
    float j_unit, harga, nilai;
    int    i;
    printf("POTONGAN HARGA SUSU\n");
    for (i = 0; i < 19; i++) printf("="); printf("\n");
    printf("jumlah unit ? ");
    scanf("%f", &j_unit);
    printf("harga unit  ? ");
    scanf("%f", &harga);
    if (j_unit <= 3)
        nilai = j_unit * harga * 0.8;
    else
        nilai = (3 * harga * 0.8) + (j_unit - 3) * harga;
    printf("total harga = %.0f\n", nilai);
    return 0;
}
```



# TIPIKAL PROGRAM C

```
1  # include <stdio.h>
2  # include <stdlib.h>
3  # define MAKS 100
```

*Preprocessor directive*

```
5  int hitung (int x) {
6      ...
7      return ...;
8  }
```

*Function* `hitung()`

```
10 int main () {
11     int    a, b;
12     ...
13     b = hitung (a);
14     ...
15     return 0;
16 }
```

*Function* `main()`

# TIPIKAL PROGRAM C

- Komentar
  - Memberi keterangan kepada program.
  - Dituliskan diantara `/*` dan `*/`.
  - Komentar tidak boleh *nested*.
  - `/*ini sebuah komentar*/`
  - `"/* ini bukan komentar melainkan string */"`
  - `/* komentar /* seperti ini */ akan menimbulkan kesalahan */`
  - Komentar satu baris `//`

# ELEMEN BAHASA C

- Karakter
- *Identifier*
  - *Keyword*
  - *Standard identifier*
  - *Programmer-defined identifier*
- Tipe data
- Konstanta
- Variabel

# KARAKTER

Jenis	Rincian
Huruf	A B C sampai dengan Z a b c sampai dengan z
Angka	0 1 2 3 4 5 6 7 8 9
Karakter khusus	! " # % & ' ( ) * + , - . / : ; < = > ? [ \ ] ^ _ {   } ~ <i>spasi</i>
Karakter extend	Tab horisontal, tab vertikal, <i>form feed, new line</i>



# IDENTIFIER

- *Identifier*
  - Identitas, pengenal, nama segala sesuatu
  - Nama instruksi, nama variabel, nama tipe data, nama function
- Ketentuan *identifiser*
  - Diawali huruf atau garis bawah (*underscore*)
  - Diikuti huruf, angka, atau garis bawah
  - Panjang maksimum 32 karakter (Borland C++)
  - Panjang tidak dibatasi (Dev-C++)
- Bahasa C bersifat *case sensitive*
  - Huruf besar dan kecil dianggap beda
  - *abc* berbeda dengan *Abc AbC ABC abC*

# IDENTIFIER

- *Identifier* yang panjang
  - luas\_persegipanjang\_berwarna\_biru
  - luas\_persegipanjang\_berwarna\_biru\_muda
- Contoh *identifier* yang benar
  - luas
  - segi3
  - bilangan\_1
- Contoh *identifier* yang salah
  - 3com                      diawali angka
  - persegi-panjang        terdapat -
  - luas lingkaran         dipisahkan spasi

# KEYWORD

- *Keyword*
  - *Identifier* yang telah diberi kegunaan tertentu dan tidak boleh diubah, hanya

1	<b>auto</b>	9	<b>double</b>	17	<b>int</b>	25	<b>struct</b>
2	<b>break</b>	10	<b>else</b>	18	<b>long</b>	26	<b>switch</b>
3	<b>case</b>	11	<b>enum</b>	19	<b>register</b>	27	<b>typedef</b>
4	<b>char</b>	12	<b>extend</b>	20	<b>return</b>	28	<b>union</b>
5	<b>const</b>	13	<b>float</b>	21	<b>short</b>	29	<b>unsigned</b>
6	<b>continue</b>	14	<b>for</b>	22	<b>signed</b>	30	<b>void</b>
7	<b>default</b>	15	<b>goto</b>	23	<b>sizeof</b>	31	<b>volatile</b>
8	<b>do</b>	16	<b>if</b>	24	<b>static</b>	32	<b>while</b>

# KEYWORD

- *Keyword*

```
int main () {  
    int i = 0;  
  
    do // OK  
        printf("%d ", i++);  
    while (i < 5);  
    return 0;  
}
```

0 1 2 3 4

```
int main () {  
    float do; // ERROR  
    ...  
}
```



# STANDARD IDENTIFIER

- *Standard Identifier*
  - *Identifier* yang telah didefinisikan *compiler* untuk kegunaan tertentu tetapi boleh

```
int main () {  
    ...  
    printf("POTONGAN HARGA SUSU\n"); // OK  
    ...  
}
```

```
int main () {  
    float printf; // OK  
    ...  
    printf("POTONGAN HARGA SUSU\n"); // ERROR  
    ...  
}
```

# PROGRAMMER-DEFINED ID

- *Programmer-Defined Identifier*
  - *Identifier* yang dibuat oleh pemrogram

```
int main () {  
    float j_unit, harga; // OK  
    ...  
    scanf("%f %f", &j_unit, &harga); // OK  
    printf("POTONGAN HARGA SUSU\n");  
    ...  
}
```

# TIPE DATA

- Tipe Data
  - Jenis data yang diolah
  - Bilangan bulat, bilangan pecahan, karakter

Type data	Keyword	Rentang Nilai (GCC)
Character	<code>char</code>	−128 s.d. 127
Short integer	<code>short</code>	−32.768 s.d. 32.767
Integer	<code>int</code>	−2.147.483.648 s.d. 2.147.483.647
Long integer	<code>long</code>	−2.147.483.648 s.d. 2.147.483.647
Long-long integer	<code>long long</code>	−9.223.372.036.854.775.808 s.d. 9.223.372.036.854.775.807
SP loating point	<code>float</code>	$1.17 \times 10^{-38}$ s.d. $3.4 \times 10^{38}$
DP floating point	<code>double</code>	$2.22 \times 10^{-308}$ s.d. $1.79 \times 10^{308}$

# TIPE DATA

- Tipe data karakter (*char*)
  - 1 *byte* = 8 *bit*
  - $2^8 = 256$  nilai  $\Rightarrow -128 \dots -1 \ 0 \ 1 \ 2 \dots 127$
- Tipe data short integer (*short*)
  - 2 *byte* = 16 *bit*
  - $2^{16} = 65536$  nilai  $\Rightarrow -32768 \dots 32767$
- Tipe data integer (*int*)
  - 4 *byte* = 32 *bit*
  - $2^{32} = 4.294.967.296$  nilai
  - $\Rightarrow -2.147.483.648 \dots 2.147.483.647$



# TIPE DATA

```
# include <stdio.h>
int main() {
    printf("char      = %d\n", sizeof(char));
    printf("short     = %d\n", sizeof(short));
    printf("int       = %d\n", sizeof(int));
    printf("long      = %d\n", sizeof(long));
    printf("float     = %d\n", sizeof(float));
    printf("double    = %d\n", sizeof(double));
    printf("long long   = %d\n", sizeof(long long));
    printf("long double = %d\n", sizeof(long double));
    return 0;
}
```

```
char      = 1
short     = 2
int       = 4
long      = 4
float     = 4
double    = 8
long long   = 8
long double = 12
```

# TIPE DATA

- *Qualifier* **unsigned** pada tipe data
  - Tidak mengakomodasi nilai negatif
  - Seluruh daya tampung untuk bilangan positif
  - *Default qualifier* adalah **signed**

Keyword	Rentang Nilai (GCC)
<b>unsigned char</b>	0 s.d. 255
<b>unsigned short</b>	0 s.d. 65.535
<b>unsigned int</b>	0 s.d. 4.294.967.295
<b>unsigned long</b>	0 s.d. 4.294.967.295
<b>Unsigned long long</b>	0 .. 18.446.744.073.709.551.615

# TIPE DATA

- *Single precision floating number (float) versus double precision floating number*

```
# include <stdio.h>
int main() {
    printf(" 123456789A123456789B\n");
    printf("%.20f\n", (float)22.0 / (float)7.0);
    printf("%.20lf\n", (double)22.0 / (double)7.0);
    return 0;
}
```

123456789A123456789B  
3.14285707473754880000  
3.14285714285714280000



# KONSTANTA

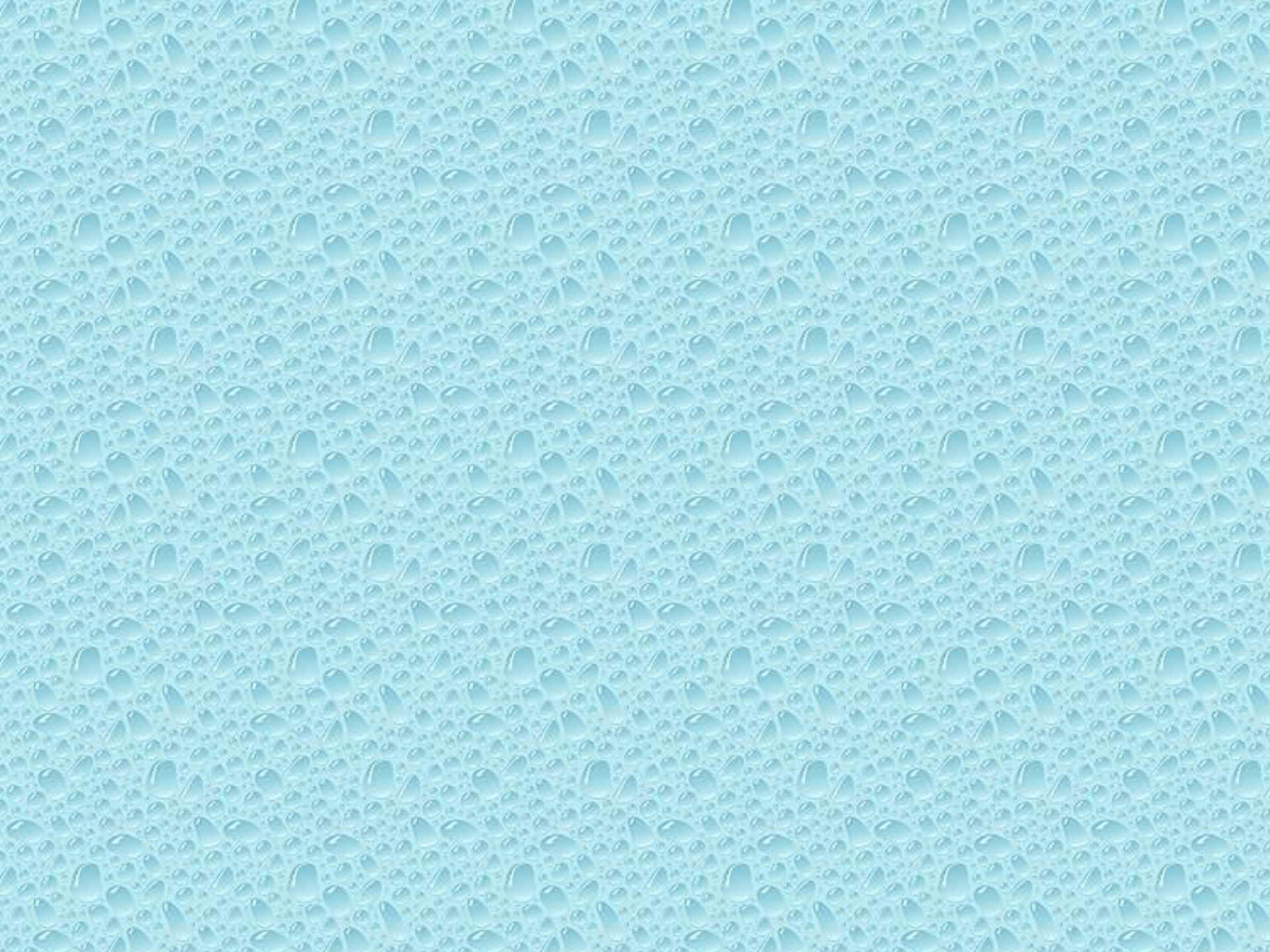
- Konstanta
  - *Integer constant*
    - Konstanta integer desimal: **125 2010**
    - Konstanta integer oktal: **07 0642**
    - Konstanta integer heksadesimal: **0x36 0Xab5**
  - *Character constant*: **'a' '1'** *escape sequence*
    - *escape sequence* diawali garis miring terbalik
  - *Floating point constant*: **3.14 1.23e4**
  - *String constant*
    - **"Dev-C++ versi 4.9.9.2"**
    - **"algoritma\npemrograman"**



# KONSTANTA

- *Escape sequence*

ES	Nilai	Karakter	Keterangan
\0	0x00	NUL	karakter dengan nilai ASCII = 0
\a	0x07	BEL	bunyi <i>speaker</i>
\b	0x08	BS	<i>backspace</i> (mundur ke kiri)
\f	0x0C	FF	<i>formfeed</i> (ganti halaman)
\n	0x0A	LF	<i>Linefeed atau newline</i> (ganti baris)
\r	0x0D	CR	<i>carriage return</i> (ke awal baris)
\t	0x09	HT	tabulator horisontal
\v	0x0B	VT	tabulator vertikal
\\	0x5C	\	karakter miring kiri
\'	0x27	'	karakter petik-tunggal
\"	0x22	"	karakter petik-ganda
\?	0x3F	?	karakter tanda tanya



# VARIABEL

- Variabel
  - Tempat menampung data
  - Harus ditentukan tipe data yang ditampung
  - Harus dideklarasikan atau didefinisikan sebelum



```
data_type var1 <= init1 >, var2 <= init2 >, ... ;
```

```
unsigned int nilai_tugas;  
int nilai_uts, nilai_uas;  
char huruf;
```

```
float jumlah = 0.0;  
char titik = '.';
```

# OPERASI dan OPERATOR

- Operasi: proses terhadap data
- Operator: simbol operasi
- Operan: data yang dioperasikan
- Berdasarkan jumlah operan terlibat
  - *unary operator*, satu operan: ++ -- unary minus
  - *binary operator*, dua operan: + - \* /
  - *ternary operator*, tiga operan: ? :
- Berdasarkan jenis operasi
  - Operator aritmetika
  - Operator relasi
  - Operator logika
  - Operator *bitwise*
  - Operator koma
  - Operator *assignment*



# OPERASI dan OPERATOR

- Operator Aritmetika

Simbol	Fungsi	Contoh
+	Penjumlahan	<code>y + 6</code>
-	Pengurangan	<code>2010 - tahunlahir</code>
*	Pekalian	<code>3.14 * diameter</code>
/	Pembagian	<code>jumlahdetik / 3600</code>
%	Modulus	<code>n % 16</code>
++	<i>Increment</i>	<code>x++ ++x</code>
--	<i>Decrement</i>	<code>z-- --z</code>

# OPERASI dan OPERATOR

- *Preincrement dan postincrement*

```
# include <stdio.h>
int main() {
    int bulat1 = 10, bulat2 = 25;
    printf("sebelum = %d\n", bulat1);
    printf("sedang   = %d\n", bulat1++);
    printf("setelah  = %d\n", bulat1);
    printf("sebelum = %d\n", bulat2);
    printf("sedang   = %d\n", ++bulat2);
    printf("setelah  = %d\n", bulat2);
    return 0;
}
```

```
sebelum = 10
sedang   = 10
setelah  = 11
sebelum = 25
sedang   = 26
setelah  = 26
```