7

FUNGSI KARAKTER DAN STRING

Pustaka (library) standard C memiliki set string dan karakater untuk menangani fungsi yang banyak dan bervariasi. Dalam bahasa C, string merupakan array karakter yang dihentikan dengan nol (null-terminated). Dalam implementasi yang standard fungsi string menuntut file header STRING.H untuk memberikan prototype mereka. Fungsi-fungsi karakter menggunakan CTYPE.H sebagai file header mereka. Merupakan hal yang mungkin bagi file-file ini untuk memiliki nama yang berbeda bila compiler tidak mengikuti standard ANSI C atau standard UNIX.

Oleh karena bahasa C tidak memiliki pengecekan batas (bound-checking) pada operasi array, maka merupakan tugas programmer untuk mencegah terjadinya berlebihannya muatan array. Sebagaimana standard ANSI C, maka bila terjadi muatan yang terlalu banyak pada array (overflow) maka gajalanya tidak didefinisikan, dengan kata lain program kita akan rusak!

Dalam bahasa C, karakter yang dapat dicetak merupakan salah satu yang dapat ditampilkan pada terminal. Karakter ini biasanya merupakan karakter antara (0x20) dan tanda tidel (0xfE). Karakter-karakter kontrol memiliki nilcai antara 0 dan 0x1F termasuk DEL (0x7F).

Berdasakan pengalaman yang sudah berlalu, argumen untuk fungsi karakter memiliki nilai integer. Akan tetapi, hanya byte yang low-order yang digunakan, fungsi karakter secara otomatis akan mengubah argumen ini menjadi **usigned char**. Namun demikian, kita bebas untuk memanggil fungsi-fungsi ini dengan argumen karakter, karena karakter secara otomatis akan diubah mejadi integer pada saat pemanggilan.

File header STRING.H mendefinisikan tipe **size_t**, yang pada dasarnya **unsigned** yang sama.

ISALNUM

```
#include "ctype.h"
int isalnum(int ch);
```

Fungsi **isalnum** akan mengebalikan nilai selain nol bila argumennya merupakan huruf alfabet atau digit. Bila karakter tidak merupakan alfanumerik, maka 0 akan dikembalikan.

Fungsi-fungsi yang berhubungan adalah **isalpha**(), **isdigit**(), **isgraph**(), **isprint**(), **ispunct**(), dan **isspace**().

ISALPHA

```
#include "ctype.h"
int isalpha(int ch);
```

Fungsi **isalpha**() akan mengembalikan nilai selain nol bila **ch** merupakan huruf alfabet; bila tidak, maka 0 yang dikembalikan. Simbol yang menunjukkan huruf pada alfabet dapat beraneka ragam dari satu bahasa ke bahasa lain. Untuk bahasa Inggris, simbol ini merupakan huruf besar dan kecil dari A sampai Z.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **iscntrl**(), **isdigit**(), **isgraph**(), **isprint**(), **ispunct**() dan **isspace**().

ISCNTRL

```
#include "ctype.h"
int iscntrl(int ch);
```

Fungsi **iscntrl**() ini akan mengembalikan nilai selain nol bila **ch** berada antara 0 dan 0x1F atau apabila **ch** sama dengan 0x7F (DEL); bila sebaliknya, maka 0 akan dikembalikan.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **isdigit**(), **isgraph**(), **isprint**(), **ispunct**(), dan **isspace**().

ISDIGIT

```
#include "ctype.h"
int isdigit(int ch);
```

Fungsi **isdigit**() adalah untuk mengembalikan nilai selain nol bila **ch** merupakan sebuah digit, yaitu '0' sampai '9'. Bila sebaliknya maka 0 akan diberikan.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **iscntrl**(), **isgraph**(), **isprint**(), **ispunct**(), dan **isspace**().

ISGRAPH

```
#include "ctype.h"
int isgraph(int ch);
```

Fungsi **isgraph**() akan mengembalikan nilai selain nol bila **ch** merupakan semua karakter selain space; bila karakter ini merupakan space (spasi) maka 0 akan dikembalikan. Meskipun karakter ini merupakan implementasi dependen, namun mereka biasanya berada pada range 0x21 sampai 0x7E.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **iscntrl**(), **isdigit**(), **isprint**(), **ispunct**(), dan **isspace**().

ISLOWER

```
#include "ctype.h"
int isloer(int ch);
```

Fungsi **islower**() akan mengembalikan nilai selain nol bila **ch** merupakan huruf kecil; bila huruf besar maka nilai 0 akan dikembalikan.

Fungsi yang berhubungan adalah **isupper**().

ISPRINT

```
#include "ctype.h"
int isprint(int ch);
```

Fungsi **isprint**() akan mengembalikan nilai selain nol bila **ch** merupakan karakter yang dapat dicetak, termasuk spasi; bila merupakan karakter yang tidak dapat dicetak maka dikembalikan nilai nol. Meskipun karakter ini merupakan implementasi dependen, namun karakter yang dapat dicetak biasanya berada pada range 0x20 sampai 0x7E.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **isalcntrl**(), **isdigit**(), **isgraph**(), **ispunct**(), dan **isspace**().

ISPUNCT

```
#include "ctype.h"
int ispunct(int ch);
```

Fungsi **ispunct**() akan mengembalikan nilai selain nol bila **ch** merupakan karakter tanda baca (punctuation); bila selain karakter ini maka akan dikembalikan nilai 0. Istilah **punctuation**, seperti yang didefinisikan oleh fungsinya, meliputi semua karakter cetak yang bukan merupakan alfanumerik, juga bukan spasi.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **isalcntrl**(), **isdigit**(), **isgraph**(), dan **isspace**().

ISSPACE

```
#include "ctype.h"
int isspace(int ch);
```

Fungsi **isspace**() akan mengembalikan nilai selain nol bila **ch** merupakan spasi, tab horisontal, form feed, dan carriage return atau karakter baris baru; bila selain mereka maka nilai 0 akan dikembalikan.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **isalcntrl**(), **isdigit**(), **isgraph**(), dan, **ispunct**().

ISUPPER

```
#include "ctype.h"
int isupper(int ch);
```

Fungsi **isupper**() akan mengembalikan nilai selain nol bila **ch** merupakan huruf besar; bila huruf kecil akan dikembalikan nilai nol.

Fungsi yang berhubungan adalah **islower**().

ISXDIGIT

```
#include "ctype.h"
int isxdigit(int ch);
```

Fungsi **isxdigit**() akan mengembalikan nilai selain nol bila **ch** merupakan digit hexadesimal; bila bukan hexadesimal maka akan dikembalikan nilai nol. Digit hexadesimal adalah antara range: A-F, a-f, atau 0-9.

Fungsi-fungsi yang berhubungan adalah **isalnum**(), **isalpha**(), **isalcntrl**(), **isdigit**(), **isgraph**(), **ispunct**(), dan **isspace**().

MEMCHR

```
#include "string.h"
void *memchr(const void *buffer, int ch, size_t count);
```

Fungsi **memchr**() akan mencari array yang ditunjukkan oleh **buffer** untuk kemunculan **ch** yang pertama kali pada karakter **count**.

Fungsi **memchr** akan mengembalikan mengembalikan pointer ke dalam kemunculan pertama **ch** dalam **buffer**, atau pointer nol bila **ch** tidak ditemukan.

MEMCMP

```
#include "string.h"
int memcmp(const void *buf1, const void *buf2, size_t
count);
```

Fungsi **memcmp**() akan membandingkan karakter **count** pertama pada array yang ditunjukkan oleh **buf1** dan **buf2**. Pembandingan ini dilakukan secara leksikografis.

Fungsi **memcmp**() akan mengembalikan integer yang diinterpretasikan sebagaimana ditunjukkan di sini.

Nilai	Arti
Kurang dari 0	**buf1** lebih kecil dari **buf2**
0	**buf1** sama dengan **buf2**
Lebih besar dari 0	**buf1** lebih besar dari **buf2**

Fungsi-fungsi yang berhubungan adalah **memchr**(), **memcpy**(), dan **strcmp**().

MEMCPY

```
#include "string.h"
void *memcpy(void *to, const void *from, size_t count);
```

Fungsi **memcpy**() akan mencopy karakter **count** dari array yang ditunjukkan oleh **from** ke dalam array yang ditunjukkan oleh **by**. Bila array ini ternyata tumpang tindih, maka gejala dari **memcpy**() tidak didefinisikan.

Fungsi **memcpy**() akan mengembalikan pointer ke dalam **to**.

Fungsi-fungsi yang berhubungan adalah **memmove**().

MEMMOVE()

```
#include "string.h"
void *memmove(void *to, const void *from, size_t count);
```

Fungsi **memmove**() adalah untuk mencopy karakter **count** dari array yang ditunjukkan oleh **from** ke dalam array yang ditunjukkan oleh **to**. Bila array ini tumpang tindih (overlap), maka copy ini akan proses copy ini akan berlangsung dengan benar, dengan menempatkan isi yang betul ke dalam **to** namun meninggalkan **from** yang telah diubah.

Fungsi **memmove**() akan mengembalikan pointer ke dalam **to**.

Fungsi yang berhubungan adalah **memcpy**().

MEMSET

```
#include "string.h"
void *memset(void *buf, int ch, size_t cont);
```

Fungsi **memset**() adalah untuk mencopy byte low-order pada **ch** ke dalam karakter **chount** pertama pada array yang ditunjuk oleh **buf**. Fungsi ini akan mengembalikan **buf**.

Pemakaian yang paling umum dari **memset**() adalah untuk memulai ruang memori untuk nilai tertentu yang diketahui.

Fungsi-fungsi yang berhubungan adalah **memcmp**(), **memcpy**(), dan **memmove**().

STRCAT

```
#include "string.h"
char *strcat(char *str1, const char *str2);
```

Fungsi **strcat**() adalah untuk menghubungkan copy **str2** dengan **str1** serta untuk menghentikan **str1** dengan nol. Penghenti nol yang sebenarnya menghentikan **str1** akan ditulis kembali oleh karakter pertama pada **str2**. String **str2** tidak dikenai operasi apapun. Bila array-array ini tumpang tindih maka gejala yang ditunjukkan **strcat**() tidak diketahui (tidak didefinisikan).

Fungsi **strcat**() akan mengembalikan **strcat1**().

Perlu diingat bahwa tidak terjadi pengecekan batas (bounds-checking), sehingga hal ini merupakan tugas programmer untuk meyakinkan bahwa **str1** cukup besar untuk menampung isi str1 itu sendiri maupun isi dari **str2**.

Fungsi-fungsi yang berhubungan adalah **strchr**(), **strcmp**(), dan **strcpy**().

STRCHR

```
#include "string.h"
char *strchr(const char *str, int ch);
```

Fungsi **strchr**() akan mengembalikan pointer ke dalam kemunculannya yang pertama pada byte low-order dari **ch** dalam string yang ditunjukkan oleh **str**. Bila tidak ditemukan kecocokkan, maka pointer akan dikembalikan.

Fungsi-fungsi yang berhubungan adalah **strpbrk**(), **strspn**(), **strstr**(), dan **strtok**().

STRCOLL

```
#include "string.h"
int strcoll(const char *str1, const char *str2);
```

Fungsi **strcoll**() akan membandingkan string yang ditunjukan oleh **str1** dengan string yang ditunjukkan oleh **str2**. Perbandingan ini akan dilakukan sesuai dengan urutan yang ditunjukkan oleh fungsi **setlocale**(). Lihat **setlocale**(*) untuk informasi yang lebih rinci.)

Fungsi **strcoll()** akan mengembalikan integer yang diinterpretasika seperti berikut.

Nilai	Arti
Kurang dari 0	**str1** lebih kecil dari **str2**
0	**str1** sama dengan **str2**
Lebih besar dari 0	**str1** lebih besar dari **str2**

Fungsi-fungsi yang berhubungan adalah **memcmp()**, **strcmp()**.

STRCMP

```
#include "string.h"
int strcmp(const char *str1, const char *str2);
```

Fungsi **strcmp()** secara leksikografi akan membandingkan dua string dan mengembalikan integer berdasarkan hasil sebagi berikut:

Nilai	Arti
Kurang dari 0	**str1** lebih kecil dari **str2**
0	**str1** sama dengan **str2**
Lebih besar dari 0	**str1** lebih besar dari **str2**

Fungsi-fungsi yang berhubungan adalah **strcmp()**, **strcmp()**, dan **strcpy** .

STRCPY

```
#include "string.h"
char *strcpy(char *str1, const char *str2);
```

Fungsi **strcpy()** digunakan untuk mengcopy isi dari **str2** ke dalam **str1**. **str2** harus merupakan pointer untuk string yang diakhiri dengan nol. Fungsi **strcpy()** akan mengembalikan pointer **str1**.

Bila **str1** dan **str2** tumpang tindih, maka akibat yang muncul tidak didefinisikan.

Fungsi-fungsi yang berhubungan adalah **memcpy**(), **strchr**(), **strcmp**(), **strncmp**().

STRCSPN

```
#include "string.h"
int strcspn(const char *str1, const *str2);
```

Fungsi **strcspn**() akan mengembalikan panjang sunstring awal dari string yang ditunjukkan oleh **str1** yang hanya tersusun atas karakter tersebut yang tidak terdapat dalam string yang ditunjuk oleh **str2**. Fungsi **strcspn**() akan mengembalikan indeks karakter pertama dalam string yang ditunjuk oleh **str1** yang sesuai dengan semua karakter dalam string yang ditunjuk oleh **str2**.

STRERROR

```
#include "string.h"
char *strerror(int errnum);
```

Fungsi **strerror** akan mengembalikan pointer ke dalam implementasi yang didefinisikan oleh string yang berhubungan dengan nilai pada **errnum**. Dalam kondisi tertentu kita harus mendefinisikan string ini.

STRLEN

```
#include "string.h"
size_t strlen(char *str);
```

Fungsi **strlen**() returns the length of the null-terminated string string pointed to by str. The null is not counted.

Fungsi yang berhubungan adalah **memcopy**(), **strchr**(), **strcmp**(), and **strncmp**().

STRNCAT

```
#include "string.h"
char *strncat(char *str1, const *str2, size_t count);
```

Fungsi **strncat**() akan hanya menghubungkan karakter **count** pada string yang ditunjuk oleh **str2** dengan string yang ditunjuk oleh **str1**. Fungsi ini juga akan mengakhiri **str1** dengan nol. Penghenti nol yang sebenarnya mengakhiri **str1** ditulis ulang oleh karkater pertama pada **str2**. **str2** tidak mengalami operasi apapun. Bila string-string ini tumpang tindih maka akibat dari proses ini tidak didefinisikan.

Fungsi **strncat**() mengembalikan **str1**.

Perlu diingat bahwa di sini tidak terdapat pengecekkan batas, oleh karena itu programmerlah yang bertanggung jawab untuk meyakinkan bahwa **str1** cukup besar untuk menampung isinya sendiri maupun isi dari **str2**.

Fungsi-fungsi yang berhubungan adalah **strcat**(), **strnchr**(), **strncmp**(), dan **strncpy**().

STRNCMP

```
#include "string.h"
int strncmp(const char *str1, const char *str2, size_t
count);
```

Fungsi **strncmp**() secara leksikografi hanya akan membandingkan karakter **count** dari dua string yang diakhiri dengan nol serta mengembalikan integer berdasarkan hasil sebagai berikut:

Nilai	Arti
Kurang dari 0	**str1** lebih kecil dari **str2**
0	**str1** sama dengan **str2**
Lebih besar dari 0	**str1** lebih besar dari **str2**

Bila terdapat karakter yang lebih sedikit dibandingkan dengan **count** dalam string, maka perbandingan ini akan diakhiri ketika pertama kali ditemukan nol.

Fungsi-fungsi yang berhubungan adalah **strcmp**(), **strnchr**(), dan **strncpy**().

STRNCPY

```
#include "string.h"
char *strncpy(char *str1, const char *str2, size_t
count);
```

Fungsi **strncpy**() digunakan untuk mengcopy karakter **count** dari string yang ditunjukkan oleh **str2** ke dalam string yang ditunjukkan oleh **str1**. **str1** harus merupakan pointer untuk string yang diakhirir dengan nol. Fungsi **strncpy**() akan mengembalikan pointer ke **str1**.

Bila **str1** dan **str2** tumpang tindih maka akibat operasi ini tidak didefinisikan.

Bila string yang ditunjukkan oleh **str2** memiliki karakter yang lebih sedikit bila dibandingkan dengan karakter **count**, maka nol akan ditambahkan pada akhir **str1** sampai karakter **count** selesai dicopy.

Kemungkinan lain adalah bila string yang ditunjuk oleh **str2** lebih panjang dari karakter **count** maka string resultant yang ditunjuk oleh **str1** tidak akan diakhiri dengan nol.

Fungsi **strncpy**() akan mengembalikan pointer ke dalam **str1**

Fungsi-fungsi yang berhubungan adalah **memcpy**(), **strchr**(), **strncat**(), dan **strncmp**().

STRPBRK

```
#include "string.H"
char *strpbrk(const char *str1, const char *str2);
```

Fungsi **strpbrk**() akan mengembalikan pointer ke dalam karakter pertama dalam string yang ditunjuk oleh **str1** yang sesuai dengan karakter apapun dalam string yang ditunjuk oleh **str2**. Piranti penghenti nol tidak termasuk di dalamnya. Bila tidak terdapat kecocokkan, maka pointer nol akan dikembalikan.

Fungsi-fungsi yang berhubungan adalah **strrchr**(), **strspn**(), **strstr**(), dan **strtok**().

STRRCHR

```
#include "string.h"
char *strrchr(const char *str, int ch)"
```

Fungsi **strrchr**() akan mengembalikan pointer ke dalam kemunculan byte low-order yang pertama dari **ch** dalam string yang ditunjuk oleh **str1**. Bila tidak ditemukan kesesuaian maka pointer nol akan dikembalikan.

Fungsi-fungsi yang berhubungan adalah **strpbrk**(), **strspn**(), **strstr**(), dan **strtok**().

STRSPN

```
#include "string.h"
size_t strspn(const char *str1, const char *str2);
```

Fungsi **strspn**() akan mengembalikan panjang dari substring awal pada string yang ditunjukkan oleh **str1** yang hanya tersusun atas karakter-karakter yang terdapat

dalam string yang ditunjuk oleh **str2**. Fungsi ini akan mengembalikan karakter pertama dalam string yang ditunjuk oleh **str1** yang tidak sesuai dengan salah satu karakter dalam string yang ditunjuk oleh **str2**.

Fungsi-fungsi yang berhubungan adalah **strpbrk**(), **strrchr**(), **strstr**(), dan **strtok**().

STRSTR

```
#include "string.h"
char *strstr(const char *str1, const char *str2);
```

Fungsi **strstr** akan mengembalikan pointer dalam kemunculannya yang pertama dalam string yang ditunjuk oleh **str1** pada string yang ditunjuk oleh **str2**. Fungsi ini akan mengembalikan pointer bila tidak terdapat kecocokkan.

Fungsi-fungsi yang berhubungan adalah **strpbrk**(), **strrchr**(), **strcspn**(), **strchr**(), **strcspn**() dan **strtok**().

STRTOK

```
#include "string.h"
char *strtok(char *str1, const char *str2);
```

Fungsi **strtok**() akan mengembalikan pointer ke dalam token berikutnya dalam string yang ditunjuk oleh **str1**. Karakter yang menyusun string yang ditunjuk oleh **str2** merupakan delimiter yang dapat mengakhiri token tersebut. Pointer nol akan dikembalikan bila tidak terdapat token yang harus dikembalikan.

Pertama kali **strtol**() dipanggil, maka **str1** sesungguhnya digunakan dalam pemanggilan tersebut. Panggilan sunsequent harus menggunakan pointer nol untuk argumen pertama.

Kita juga dapat menggunakan set yang berbeda untuk masing-masing panggilan ke dalam **strtok**().

Fungsi-fungsi yang berhubungan adalah **strpbrk**(), **strrchr**(), **strcspn**(), **strchr**(), dan **strspn**().

STRXFRM

```
#include "string.h"
size_t strxfrm(char *str1, const char *str2, size_t
count);
```

Fungsi **strxfrm**() akan mengubah karakter **count** pertama pada string yang ditunjuk oleh **str2** sehingga mereka dapat dipakai oleh fungsi **strcmp**(). Fungsi ini juga akan menempatkan hasil ke dalam string yang ditunjuk oleh **str1**. Setelah transfomasi ini, maka hasil dari **strcmp**() yang menggunakan **str1** dan **strcoll**() yang menggunakan string sesungguhnya yang ditunjuk oleh **str2** akan merupakan hal yang sama. Fungsi ini terutama digunakan dalam lingkungan bahasa asing yang tidak menggunakan urutan ASCII.

Fungsi **strxfrm**() akan mengembalikan panjang string yang telah diubah.

Fungsi yang berhubungan adalash **strcoll**().

TOLOWER

```
#include "stype.h"
int tolower(int ch);
```

Fungsi **tolower**() akan mengembalikan padanan huruf kecil pada **ch** bila **ch** berupa huruf; bila selain huruf maka **ch** akan dikembalikan sesuai keadaan sebelumnya.

Fungsi yang berhubungan adalah **toupper**().

TOUPPER

```
#include "ctype.h"
int toupper(int ch);
```

Fungsi **toupper**() akan mengmbalikan padanan huruf besar dari **ch** bila **ch** berupa huruf; bila dia tidak berupa huruf maka **ch** akan dikembalikan seperti semula.

Fungsi yang berhubungan adalah **tolower**().