



POLITEKNIK ELEKTRONIKA NEGERI SURABAYA



DIKTISAINTEK
BERDAMPAK

KAMPUS
INOVASI

POINTER

Konsep Pemrograman

Yunia Ikawati

Referensi: Umi Sa'adah, Entin Martiana Kusumaningtyas, Tri Hadiah Muliawati

Teknik Informatika-PENS





DIKTISAINTEK
BERDAMPAK

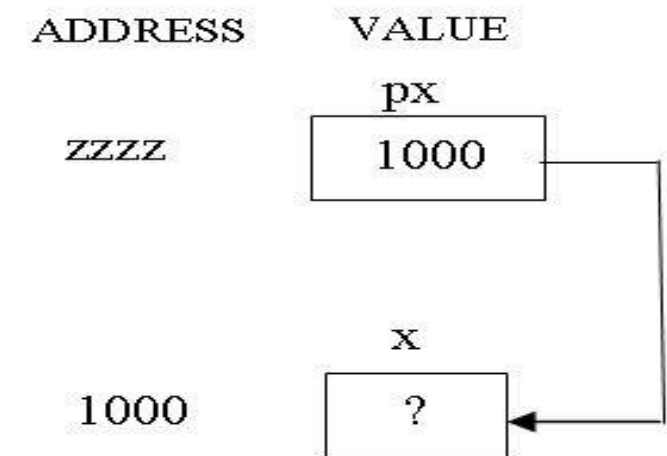
KAMPUS
INOVASI

Overview

- Konsep Dasar Pointer
- Deklarasi Variabel Pointer
- Mengatur Pointer agar Menunjuk ke Variabel Lain
- Akses INDIRECT melalui pointer

Konsep Dasar Pointer

- Pointer adalah variabel yang khusus digunakan untuk menampung address.
- Pointer sering dikatakan sebagai variabel yang menunjuk ke obyek/variabel lain.
- Kenyataan sebenarnya, variabel pointer berisi alamat dari suatu obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer).
- Misalnya:
 - **px** adalah variabel pointer
 - **x** adalah variabel yang ditunjuk oleh **px**.
 - Kalau **x** berada pada alamat memori 1000, maka **px** akan berisi 1000



Deklarasi Variabel Pointer

- Pointer dideklarasikan dengan bentuk sbb:

`tipe_data *nama_variabel;`

- **tipe_data** dapat berupa sembarang tipe yang sudah dibahas pada bab-bab sebelumnya, maupun bab-bab berikutnya.
- **nama_variabel** adalah nama dari variabel pointer.
- Jika dideklarasikan :
`int *px;`
 - nama var = `px`; tipenya = *pointer to int*
 - menyatakan bahwa `px` adalah variabel pointer yang menunjuk ke suatu data tertentu yang bertipe *int*

Mengarahkan Pointer ke Variabel Lain

- Mula-mula pointer diisi dengan alamat dari variabel yang akan ditunjuk.
- Untuk menyatakan alamat dari suatu variabel, digunakan operator `&` (operator alamat, bersifat *unary*) yang ditempatkan di depan nama variabel.
- Jika dideklarasikan :

```
int *px, x = 10;
```

Maka `&x` berarti “alamat dari variabel **x**”.

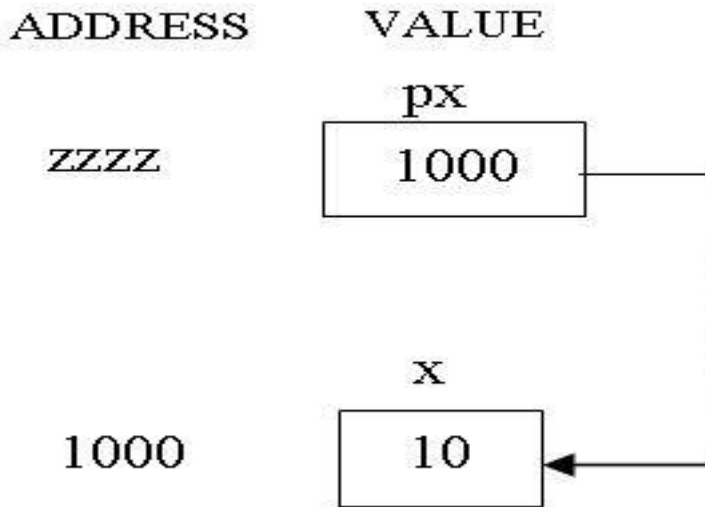
```
px = &x;
```

berarti bahwa `px` diberi nilai berupa alamat dari variabel **x**.

- Setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa **px** menunjuk ke variabel **x**.

Mengarahkan Pointer ke Variabel Lain

- Hasilnya :



- Suatu lokasi/address yg telah ditunjuk oleh sebuah pointer, maka lokasi tsb value-nya bisa diakses baik secara DIRECT maupun INDIRECT melalui pointernya**

Akses INDIRECT melalui pointer

- Pengaksesan tak langsung dilakukan dengan menggunakan operator *indirection* (tak langsung) berupa simbol ***** (bersifat *unary*).

`*px`

menyatakan “nilai atau value dari variabel/data yang ditunjuk oleh pointer **px**” .

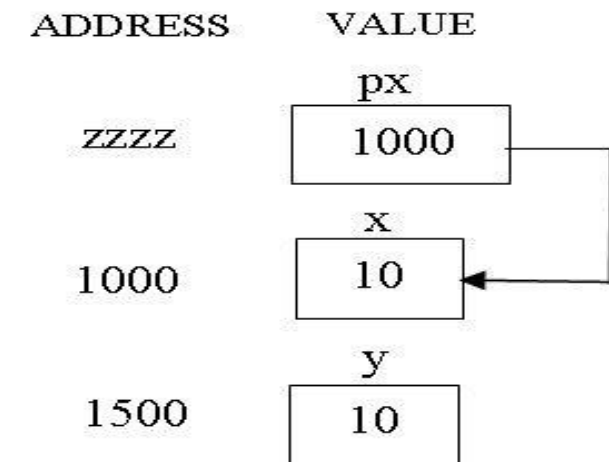
- Contoh

```
int *px, x = 10, y;
```

```
px = &x;
```

```
y = *px;
```

maka `y` akan berisi 10 yaitu
nilai yang sama dengan nilai `x`



Akses INDIRECT melalui pointer

ATURAN PEMBACAAN

- **&x** = alamat/address dari variabel **x**
- ***px** = nilai/value yang ada pada address/lokasi **px**
 - Baca ulang dengan cara mengganti **px** dengan value yang ada pada **px** (dalam contoh di atas **px** berisi 1000)
 - merupakan cara akses indirect

Keterangan:

- **x** adalah nama variabel yang bertipe *int*
- **px** adalah nama variabel yang bertipe *pointer to int*

Contoh

```
#include <stdio.h>
main()
{
    int y, x = 87, *px;

    px = &x;
    y = *px;

    printf("Alamat x      = %p\n", &x);
    printf("Isi px        = %p\n", px);
    printf("Isi x          = %d\n", x);
    printf("*px          = %d\n", *px);
    printf("Isi y          = %d\n", y);
}
```

```
C:\ "G:\Kampus\Programming ..."
Alamat x      = 0012FF78
Isi px        = 0012FF78
Isi x          = 87
*px          = 87
Isi y          = 87
Press any key to continue
```



Akses INDIRECT melalui pointer

- Tipe variabel pointer dan tipe data yang ditunjukknya harus sejenis. Bila tidak sejenis, akan terjadi hasil yang tidak diinginkan



Contoh



```
main() {  
    float *pu, nu;  
    double u = 158.0;  
  
    pu = &u;  
    nu = *pu;  
  
    printf("Alamat dari u = %p\n", &u);  
    printf("Isi pu          = %p\n", pu);  
    printf("Isi u          = %lf\n", u);  
    printf("Nilai yang ditunjuk oleh pu = %f\n", *pu);  
    printf("Nilai nu          = %f\n", nu);  
}
```

```
C:\TEMP\String\ptr\Debug\ptr.exe  
Alamat dari u = 0012FF70  
Isi pu          = 0012FF70  
Isi u          = 158.000000  
Nilai yang ditunjuk oleh pu = 0.000000  
Nilai nu          = 0.000000  
Press any key to continue
```

Latihan

Untuk setiap program di bawah ini,

- gambarkan ilustrasi alokasi memori dari setiap baris pernyataan yang diproses
- perkirakan hasil eksekusinya

```
1. main(){
    int y, x = 87;
    int *px;

    px = &x;
    y = *px;
    printf("Alamat x      = %p\n", &x);
    printf("Isi px       = %p\n", px);
    printf("Isi x        = %d\n", x);
    printf("Nilai yang ditunjuk oleh px = %d\n", *px);
    printf("Nilai y       = %d\n", y);
}
```



Latihan

```
2. main() {  
    int z = 20, s = 30, *pz, *ps;  
    pz = &z;  
    ps = &s;  
    *pz += *ps;  
    printf("z = %d\n", z);  
    printf("s = %d\n", s);  
}
```

```
3. main() {  
    char c = 'Q';  
    char *cp = &c;  
    printf("%c %c\n", c, *cp);  
    c = '/';  
    printf("%c %c\n", c, *cp);  
    *cp = '(';  
    printf("%c %c\n", c, *cp);  
}
```



Latihan

```
4. main() {  
    int x = 1, y = 2, *ip;  
    ip = &x;  
    y = *ip;  
    *ip = 3;  
    printf("x = %d, y = %d", x, y);  
}
```

```
5. main(){  
    int i1, i2, *p1, *p2;  
    i1 = 9;  
    p1 = &i1;  
    i2 = *p1 / 2 - 2 * 3;  
    p2 = p1;  
    printf("i1=%d,i2=%d,*p1=%d,*p2=%d\n",i1,i2,*p1,*p2);  
}
```



Latihan

```
6. main() {  
    int count = 10, *temp, sum = 7;  
    temp = &count;  
    *temp = 32;  
    temp = &sum;  
    *temp = count;  
    sum = *temp * 4;  
    printf("count = %d, *temp = %d, sum = %d\n", count, *temp, sum );  
}
```

```
7. main(){  
    int count = 13, sum = 9, *x, *y;  
    x = &count;  
    *x = 27;  
    y = x;  
    x = &sum;  
    *x = count;  
    sum = *x / 2 * 3;  
    printf("count = %d, sum = %d, *x = %d, *y = %d\n",  
count, sum, *x, *y);  
}
```

Referensi

1. Brian W. Kernighan, Dennis M. Ritchie (2012): The C Programming Language : Ansi C Version 2 Edition, PHI Learning
2. Byron Gottfried (2010) : Programming with C, Tata McGraw - Hill Education
3. Kochan Stephen (2004) : Programming in C, 3rd Edition, Sams
4. K. N. King (2008) : C Programming: A Modern Approach, 2nd Edition, W.
 - W. Norton & Company
5. Abdul Kadir (2012) : Algoritma & Pemrograman Menggunakan C & C++, Andi Publisher, Yogyakarta

6. <http://www.gdsw.at/languages/c/programming-bbrowm/>

7. <https://www.netanikode.com/tutorial/c/>



DIKTISAINTEK
BERDAMPAK



POINTER 2

Konsep Pemrograman

Yunia Ikawati

Referensi: Umi Sa'adah, Entin Martiana Kusumaningtyas, Tri Hadiah Muliawati

Teknik Informatika-PENS



DIKTISAINTEK
BERDAMPAK

KAMPUS
INOVASI

Overview

- Pointer to array
- Pointer to string
- Array of pointer
- Pointer to pointer

Pointer to Array

- Hubungan antara pointer dan array pada C sangatlah erat.
- Ingat bahwa sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer

array yang dituliskan tanpa kurung sikunya $\Leftrightarrow$ alamat dr elemen pertama (indeks ke-0) dr array tsb.

Pointer to Array

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
int tgl_lahir[] = {16, 8, 2003};
```

```
int *ptgl;
```

```
ptgl = tgl_lahir;
```

```
printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
```

```
printf("Nilai dari tgl_lahir[0] = %d\n", tgl_lahir[0]);
```

```
}
```

`ptgl = tgl_lahir;`
artinya sama dengan
`ptgl = &tgl_lahir[0];`
`ptgl` adalah pointer to array of integer

```
C:\K:\NEWPRAKT\string\gets\Debug\gets.exe
Nilai yang ditunjuk oleh ptgl = 16
Nilai dari tgl_lahir[0] = 16
```

Pointer to Array

```
#include <stdio.h>
main()
{
    int tgl_lahir[] = {16, 8, 2003};
    int *ptgl, i;

    ptgl = tgl_lahir;
    printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
    for (i=0; i<3; i++)
        printf("Nilai dari tgl_lahir[i] = %d\n", *(ptgl+i));
}
```

```
C:\K:\NEWPRAKT\string\gets\Debug\gets.exe
Nilai yang ditunjuk oleh ptgl = 16
Nilai dari tgl_lahir[i] = 16
Nilai dari tgl_lahir[i] = 8
Nilai dari tgl_lahir[i] = 2003
```

Untuk setiap iterasi, indeks i dinaikkan sebesar
: **1 x size** dari satu elemen array

Pointer to Array

```
#include <stdio.h>
main()
{
    int tgl_lahir[] = {16, 4, 1974};
    int i, *ptgl;

    ptgl = tgl_lahir;

    printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
    for (i=0; i<3; i++)
        printf("Nilai dari tgl_lahir[%d] = %d\n", i, *ptgl++);
}
```

```
C:\K:\NEW\PRAKT\string\gets\Debug\gets....
Nilai yang ditunjuk oleh ptgl = 16
Nilai dari tgl_lahir[0] = 16
Nilai dari tgl_lahir[1] = 4
Nilai dari tgl_lahir[2] = 1974
```

Post-increment, setiap kali iterasi address pd `ptgl` dinaikkan sebanyak **1 X size** dari satu elemen array

Pointer to Array

- Untuk mengarahkan pointer `ptgl` agar menunjuk ke elemen pertama dari array `tgl_lahir`:

```
ptgl = tgl_lahir;
```

INGAT !!

array `tgl_lahir` dituliskan tanpa kurung sikunya $\Leftrightarrow$ `&tgl_lahir[0]`

- Menampilkan data pada elemen pertama tsb dengan cara : `*ptgl`
- Untuk mengakses elemen-elemen berikutnya, gunakan looping dengan salah satu cara sbb
 - `*(ptgl+i)`
 - `*ptgl++`

Setiap kali iterasi address pd `ptgl` dinaikkan sebanyak **1 X size** dari satu elemen array

Pointer to String

- String bukanlah sebuah tipe data baru dalam C, melainkan merupakan sekumpulan karakter, sehingga string = *array of char*
- Variabel array yang dituliskan tanpa kurung sikunya
↔ alamat/address dari array tsb pada indeks ke-0



Pointer to String

```
#include <stdio.h>
main() {
    char kota[] = "Sby", *pkota;

    pkota = kota;           //pkota = &kota[0]
    printf("String yang ditunjuk oleh pkota = ");
    printf("%s\n", pkota);   //printf("%s\n", kota);
}
```

```
#include <stdio.h>
main() {
    //pkota adl pointer yg menunjuk konstanta string "Surabaya"
    char *pkota = "Sby";

    printf("String yang ditunjuk oleh pkota = ");
    printf("%s\n", pkota);
}
```



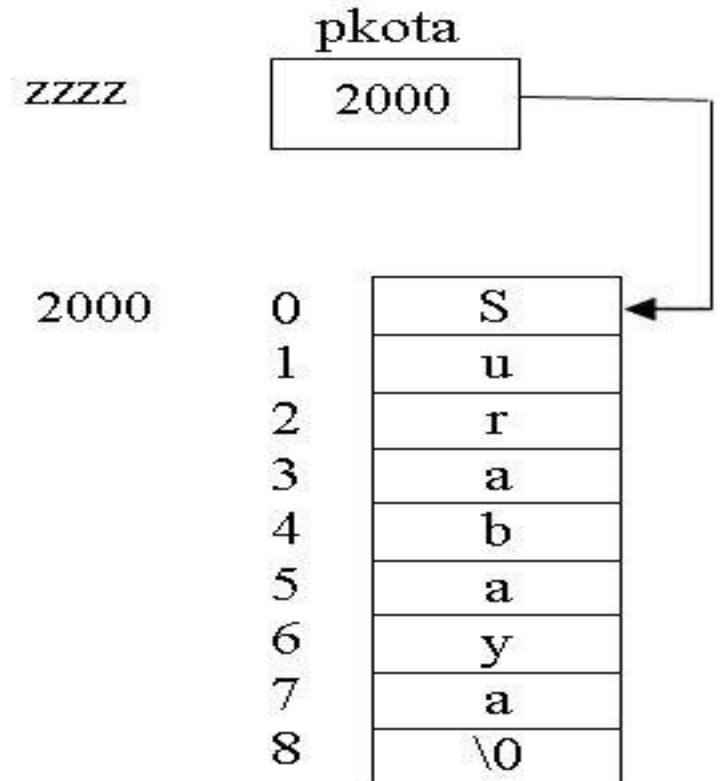
Pointer to String

- Pada program di atas,

```
char *pkota = "Surabaya";
```

akan menyebabkan kompiler mengalokasikan variabel `pkota` sebagai variabel pointer yang menunjuk ke obyek bertipe *char* dan menempatkan konstanta "Surabaya" dalam suatu memori

- kemudian pointer `pkota` akan menunjuk ke lokasi karakter 'S' dari string "Surabaya"



DIKTISAINTEK
BERDAMPAK

KAMPUS
INOVASI

Array of Pointer

- Suatu array bisa digunakan untuk menyimpan sejumlah pointer.

- Jika dideklarasikan :

```
char *nama_hari[10];
```

merupakan pernyataan untuk mendeklarasikan *array of pointer to char*.

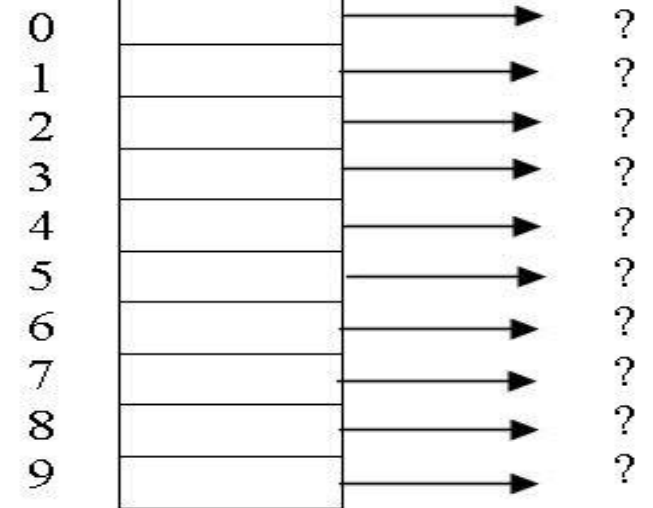
- Array `nama_hari` terdiri dari 10 elemen berupa pointer yang menunjuk ke data bertipe *char*.

ADDRESS

VALUE

nama_hari

zzzz



Array of Pointer

- Array pointer bisa diinisialisasi sewaktu pendeklarasian.
- Jika dideklarasikan:

```
char *namahari[] =  
    {"Senin",  
     "Selasa",  
     "Rabu",  
     "Kamis",  
     "Jumat",  
     "Sabtu",  
     "Minggu"};
```

- Pada contoh ini :
 - namahari[0] berisi alamat/pointer yang menunjuk ke string "Senin".
 - namahari[1] berisi alamat/pointer yang menunjuk ke string "Selasa".
 - namahari[2] berisi alamat/pointer yang menunjuk ke string "Rabu".
 - dan seterusnya

Pointer to Pointer

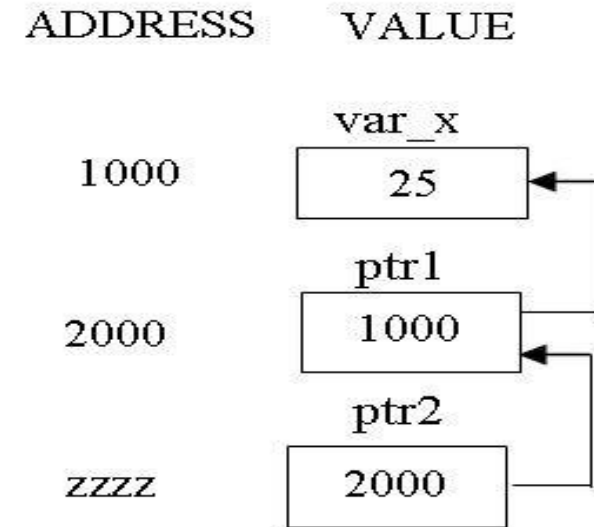
- Suatu pointer bisa menunjuk ke pointer yang lain

- Jika dideklarasikan :

```
int var_x = 25, *ptr1, **ptr2;
```

- `var_x` adalah variabel bertipe *int*.
- `ptr1` adalah variabel bertipe *pointer to int* □
pointer yang menunjuk ke sebuah data bertipe *int*
- `ptr2` adalah variabel bertipe *pointer to pointer to int* □ pointer yang menunjuk ke *pointer to int* (itulah sebabnya deklarasinya berupa `int **ptr2;`)
- Agar `ptr1` menunjuk ke variabel `var_x` dan `ptr2` menunjuk ke `ptr1`, instruksinya sbb :

```
ptr1 = &var_x;  
ptr2 = &ptr1;
```





Pointer to Pointer

```
#include <stdio.h>

main() {
    int var_x = 273, *ptr1, **ptr2;

    ptr1 = &var_x;
    ptr2 = &ptr1;

    printf("Alamat var_x = %p\n", &var_x);
    printf("Alamat ptr1 = %p\n", &ptr1);
    printf("Isi var_x = %d\n", var_x);
    printf("Isi ptr1 = %p\n", ptr1);
    printf("Isi ptr2 = %p\n", ptr2);
    printf("\n*ptr1 = %d\n", *ptr1);
    printf("**ptr2 = %d\n", **ptr2);
}
```

```
C:\ "G:\Kampus\Programming 2\..."
Alamat var_x = 0012FF7C
Alamat ptr1 = 0012FF78
Isi var_x = 273
Isi ptr1 = 0012FF7C
Isi ptr2 = 0012FF78

*ptr1 = 273
**ptr2 = 273
Press any key to continue
```

Suatu lokasi/address yang telah ditunjuk oleh sebuah pointer, maka lokasi tsb value-nya bisa diakses secara DIRECT maupun INDIRECT melalui pointernya



DIKTISAINTEK

KAMPUS
INOVASI

Latihan

Untuk setiap program di bawah ini,

- gambarkan ilustrasi alokasi memori dari setiap baris pernyataan yang diproses
- perkirakan hasil eksekusinya

1. Menukarkan isi 2 string tanpa pemakaian pointer

```
#include <stdio.h>
#include <string.h>
#define PANJANG 20
char nama1[PANJANG] = "AHMAD";
char nama2[PANJANG] = "RIFDA";
main(){
    char namax[PANJANG];
    puts("SEMULA : ");
    printf("nama1 --> %s\n", nama1);
    printf("nama2 --> %s\n", nama2);
    strcpy(namax, nama1);
    strcpy(nama1, nama2);
    strcpy(nama2, namax);
    puts("KINI : ");
    printf("nama1 --> %s\n", nama1);
    printf("nama2 --> %s\n", nama2);
}
```



Latihan

2. Menukarkan isi 2 string dengan fasilitas pointer.

```
#include <stdio.h>
char *nama1 = "AHMAD";
char *nama2 = "RIFDA";

main() {
    char *namax;

    puts("SEMULA : ");
    printf("nama1 --> %s\n", nama1);
    printf("nama2 --> %s\n", nama2);
    namax = nama1;
    nama1 = nama2;
    nama2 = namax;
    puts("KINI : ");
    printf("nama1 --> %s\n", nama1);
    printf("nama2 --> %s\n", nama2);
}
```

Referensi

1. Brian W. Kerninghan, Dennis M. Ritchie (2012): The C Programming Language : Ansi C Version 2 Edition, PHI Learning
2. Byron Gottfried (2010) : Programming with C, Tata McGraw - Hill Education
3. Kochan Stephen (2004) : Programming in C, 3rd Edition, Sams
4. K. N. King (2008) : C Programming: A Modern Approach, 2nd Edition, W. W. Norton & Company
5. Abdul Kadir (2012) : Algoritma & Pemrograman Menggunakan C & C++, Andi Publisher, Yogyakarta
6. <http://www.gdsw.at/languages/c/programming-bbrowne/>
7. <http://www.petanikode.com/tutorial/c/>
8. <http://www.cprogramming.com/tutorial/c-tutorial.html>



bridge to the future

<http://www.eepis-its.edu>