



# Pendalaman String (C++)

Tim Olimpiade Komputer Indonesia

# Bagian 1

## Pengolahan String



# String pada C++

Seperti yang disinggung sebelumnya, terdapat dua macam string yang dapat digunakan di C++.

1. `cstring`: string yang diwariskan dari bahasa C, yang merupakan leluhur bahasa C++.
2. `std string`: string yang dimiliki oleh C++.

Kita akan membahasnya satu per satu.



## Bagian 2

### **cstring**



## Pengenalan `cstring`

- Dalam bahasa C dan C++, `cstring` dibuat menggunakan " *null terminated array of char*".
- Penggunaannya sesederhana membuat array `char`.
- Karena berasal dari bahasa C, `cstring` dapat berinteraksi dengan `scanf` dan `printf` secara langsung.

---

```
#include <cstdio>
```

```
char s[1001];
```

```
int main() {  
    scanf("%s", s);  
    printf("%s\n", s);  
}
```

---



## Representasicstring

- Misalkan kita memasukkan "Pak Dengklek" pada program sebelumnya.
- Array `s` akan berisi karakter-karakter penyusun "Pak Dengklek", diakhiri dengan karakter `'\0'`.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
P	a	k		D	e	n	g	k	l	e	k	\0			...

- Karakter `'\0'` digunakan oleh C/C++ untuk menandai akhir dari string.
- Jadi, meskipun terdapat 1001 elemen pada `s`, C/C++ tahu string yang direpresentasikan hanya sampai elemen ke-11 saja.



# Representasi cstring

- Perhatikan bahwa s hanya dapat menampung paling banyak 1000 karakter.
- Meskipun memiliki 1001 elemen, sebuah elemen sudah dipesan untuk menempatkan karakter \0.
- Apabila string yang dibaca lebih dari 1000 karakter, program tidak mengalami error. Namun, program akan bekerja dengan tidak terprediksi.
- Pastikan ukuran array dibuat sebesar ukuran maksimal masukan yang mungkin.



# Pengolahan cstring

- Seperti array pada umumnya, Anda dapat mengolah setiap elemennya secara independen.
- Sebagai contoh, program berikut mengubah setiap karakter dari string yang dibaca menjadi 'a'.

---

```
int main() {  
    scanf("%s", s);  
    for (int i = 0; s[i] != '\0'; i++) {  
        s[i] = 'a';  
    }  
    printf("%s\n", s);  
}
```

---





# Mencari Panjang cstring

- Kita dapat menggunakan fungsi `strlen` untuk mencari panjang suatu cstring.
- Fungsi ini disediakan oleh STL "cstring".

## Contoh:

```
#include <stdio>
#include <cstring>

char s[1001];

int main() {
    scanf("%s", s);
    printf("%d\n", strlen(s));
}
```



## Penggunaan strlen

- Fungsi strlen memiliki kompleksitas  $O(N)$ , dengan  $N$  adalah panjang dari string masukan.
- Penyebabnya adalah strlen sebenarnya mencari di indeks keberapakah `\0` berada.
- Apabila hendak digunakan dalam perulangan, simpan dulu panjangnya ke dalam suatu variabel untuk menghindari pemanggilan yang berulang-ulang.

Contoh:

---

```
// Buruk,  $O(N^2)$ 
for (int i = 0; i < strlen(s); i++) {
    printf("%c", s[i]);
}
```

```
// Baik,  $O(N)$ 
int len = strlen(s);
for (int i = 0; i < len; i++) {
    printf("%c", s[i]);
}
```



## Membandingkan cstring

- Untuk membandingkan 2 cstring `s` dan `t` secara leksikografis, gunakan `strcmp(s,t)`.
- Nilai kembaliannya memiliki arti sebagai berikut:
  - Negatif, artinya `s` lebih awal dari `t`.
  - Nol, artinya `s` sama dengan `t`.
  - Positif, artinya `s` lebih akhir dari `t`.
- Kompleksitasnya  $O(N)$ , dengan  $N$  adalah panjang string terkecil yang dibandingkan.

Contoh penggunaan:

---

```
#include <stdio>
#include <cstring>

char s[1001];
char t[1001];

int main() {
    scanf("%s %s", s, t);
    printf("%d\n", strcmp(s, t));
}
```



## Mengisi Array

- Untuk mengisi array arr dengan x, gunakan `memset(arr, x, sizeof(arr))`.
- Nilai x terbatas pada tipe data char, atau angka di antara -128 sampai 127 saja.
- Fungsi ini biasanya juga dimanfaatkan untuk menginisialisasi array bilangan dengan 0 atau -1.

Contoh:

---

```
#include <stdio>
#include <cstring>

char s[1001];
int arr[101];

int main() {
    memset(s, 'x', sizeof(s));
    memset(arr, -1, sizeof(arr));
    printf("%c %d\n", s[0], arr[0]);
}
```



## Fungsi cstring Lainnya

- Terdapat beberapa fungsi cstring lain seperti `strcpy`, `strncpy`, dan `strstr`.
- Fungsi-fungsi tersebut tidak dibahas pada materi ini.
- Anda dapat mempelajarinya di dokumentasi C/C++ lebih lanjut, seperti di <http://www.cplusplus.com>.



## Bagian 3

### **std string**



# Representasi string

- String pada C++ direpresentasikan dengan struktur array dinamis.
- Ukurannya dapat berubah sesuai dengan ukuran string.
- Oleh sebab itu, Anda tidak perlu mendeklarasikan ukuran terbesar yang mungkin.
- Anda dapat mengakses dan mengubah nilai karakter dari suatu indeks string secara independen.



# Mencari Panjang string

- Untuk mencari panjang dari string `s`, cukup panggil `s.length()`.
- Kompleksitas pemanggilannya adalah  $O(1)$ .

---

```
// O(N)
for (int i = 0; i < s.length(); i++) {
    printf("%c", s[i]);
}
```

---





# Mencari Substring

- Untuk mencari posisi suatu substring `t` dari string `s`, gunakan `s.find(t)`.

Contoh:

---

```
#include <cstdio>
#include <string>
using namespace std;

int main() {
    string s = "Pak Dengklek berternak";
    string t1 = "Dengklek";
    string t2 = "pak";
    string t3 = "klek";

    printf("%d\n", s.find(t1)); // 4
    printf("%d\n", s.find(t2)); // -1 (tak ditemukan)
    printf("%d\n", s.find(t3)); // 8
}
```

---



# Mengambil Substring

- Untuk mengambil substring dari indeks *i* sebanyak *n* karakter dari string *s*, gunakan `s.substr(i, n)`.

Contoh:

---

```
#include <cstdio>
#include <string>
using namespace std;

int main() {
    string s = "Pak Dengklek berternak";

    printf("%s\n", s.substr(0, 6).c_str()); // Pak De
    printf("%s\n", s.substr(2, 1).c_str()); // k
}
```

---



# Menghapus Substring

- Untuk menghapus substring dari indeks *i* sebanyak *n* karakter dari string *s*, gunakan `s.erase(i, n)`.

Contoh:

---

```
#include <cstdio>
#include <string>
using namespace std;

int main() {
    string s = "Pak Dengklek berternak";
    s.erase(1, 3);

    printf("%s\n", s.c_str()); // PDengklek berternak
}
```

---



# Menyisipkan String

- Untuk menyisipkan string t ke string s bermula di indeks i, gunakan `s.insert(i, t)`.

Contoh:

---

```
#include <cstdio>
#include <string>
using namespace std;

int main() {
    string s = "Pak Dengklek berternak";
    string t = "dan Bu ";

    s.insert(4, t);
    printf("%s\n", s.c_str()); // Pak dan Bu Dengklek
                                berternak
}
```

---



# Operasi Tambahan: Penempelan String

- Pada std string, hal ini dapat dilakukan cukup dengan operasi '+', layaknya operasi numerik.

Contoh:

---

```
#include <cstdio>
#include <string>
using namespace std;

int main() {
    string s = "Pak";
    string t = "Dengklek";

    string gabung = s + t;
    printf("%s\n", gabung.c_str()); // PakDengklek
}
```

---



# Operasi Char

- Setiap karakter dari string memiliki tipe char.
- Kita dapat melakukan operasi penambahan atau pengurangan pada char, yang akan dioperasikan pada kode ASCII-nya.

Contoh:

---

```
#include <stdio>
#include <string>
using namespace std;

int main() {
    string s = "abc";
    s[0]++;
    s[1] += 2;
    s[2] -= 2;
    printf("%s\n", s.c_str()); // bda
}
```

---



## Operasi Char (lanj.)

- Operasi ini dapat digunakan untuk mengubah karakter suatu string.
- Operasi yang umum adalah mengubah dari huruf kecil ke besar, dengan cara mengurangi char dengan selisih antara ASCII 'a' dengan 'A'.
- Cara sebaliknya akan mengubah dari huruf besar ke huruf kecil.
- Hafalkan ASCII 'a' adalah 97, dan 'A' adalah 65.

```
#include <stdio>
#include <string>
using namespace std;

int main() {
    string s = "toki";
    for (int i = 0; i < s.size(); i++) {
        s[i] -= 'a' - 'A';
    }
    printf("%s\n", s.c_str()); // TOKI
}
```



## Selanjutnya...

- Pembelajaran kalian tentang C++ sudah cukup untuk bisa menuliskan algoritma-algoritma kompleks.
- Berikutnya kita akan mempelajari hal-hal yang lebih berkaitan dengan **algoritma**, bukan sekedar belajar bahasa.

