

Binary File

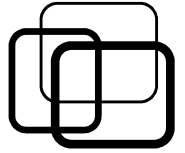
Inst. Nguyễn Minh Huy

Contents



- Dynamic string.
- Binary file.
- `main()` arguments.

Contents



- **Dynamic string.**
- Binary file.
- `main()` arguments.

Dynamic string



■ Pointer as string:

■ String = array of chars + '\0'

```
char s1[ 6 ] = { 'H', 'e', 'l', 'l', 'o', '\0' }; s1
```

H	e	l	l	o	\0
---	---	---	---	---	----

```
char s2[ ] = "Hello"; s2
```

H	e	l	l	o	\0
---	---	---	---	---	----

■ Dynamic string:

- Use dynamic array of chars.
- Flexible size.
- Declaration: `char *<string pointer>;`

```
char *s3 = new char[6]; s3
```

27	→	?	?	?	?	?	?
----	---	---	---	---	---	---	---

```
//...  
delete [ ]s3;
```

Dynamic string



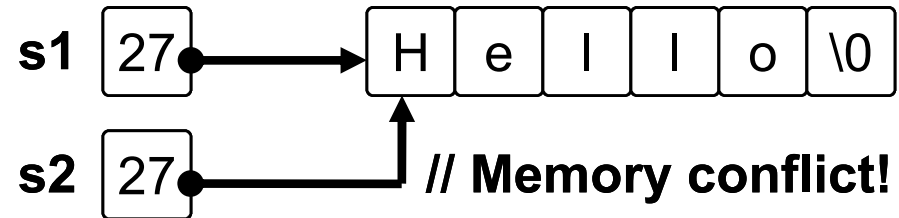
■ Library <string.h>:

■ Copy string:

- Do not use “=” to copy a string!!

```
char s1[] = "Hello";
```

```
char *s2 = s1;
```

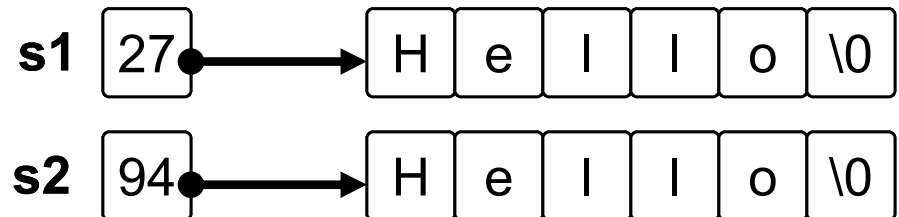


- Steps to copy string:

- Step 1: allocate new string.
- Step 2: use **strcpy** to copy string content.

```
char s1[] = "Hello";
```

```
char *s2 = new char[ strlen(s1) + 1 ];  
strcpy( s2, s1 );
```



Dynamic string



■ Library <string.h>:

■ Duplicate string:

- Syntax: **strdup**(<source string>);
- Return: new string copied from source string.
- Memory of new string must be de-allocated.

```
char s1[ ] = "Hello";  
char *s2 = strdup(s1);
```

```
// Same as....
```

```
// char *s2 = new char[ strlen(s1) + 1 ];
```

```
// strcpy( s2, s1 );
```

```
free(s2);
```

Dynamic string



■ Library <string.h>:

■ Compare string:

- Syntax: **strcmp**(<string 1>, <string 2>);
- Return: 0 (equal), 1 (greater), -1 (less).
- Compare based on dictionary order.

```
char s1[ ] = "abc";
```

```
char s2[ ] = "abaab";
```

```
char s3[ ] = "Abc";
```

```
int r1 = strcmp(s1, s2);           // r1 = 1.
```

```
int r2 = strcmp(s3, s1);           // r2 = -1.
```

```
int r3 = strcasecmp(s3, s1);       // r3 = 0, GCC compiler.
```

```
int r4 = stricmp(s3, s1);          // r4 = 0, MSVC compiler.
```

Dynamic string



■ Library <string.h>:

■ Join string:

- Syntax: **strcat**(<dest string>, <source string>);
- Join source string into dest string.
- Dest string must have enough memory!!

```
char s1[ ] = "Hello";  
char s2[ ] = "World";  
int len1 = strlen(s1);  
int len2 = strlen(s2);  
char *s3 = new char[ len1 + len2 + 1 ];
```

```
strcat(s3, s1);      // Join s1 into s3.  
strcat(s3, s2);      // Then, join s2 into s3.  
// More efficient way...  
strcpy(s3, s1);  
strcpy(s3 + len1, s2);
```


Dynamic string



■ Library <string.h>:

■ Find sub string:

- Syntax: **strstr**(<source string>, <sub string>);
- Return: address of found sub string, or NULL (not found).

```
char s1[ ] = "Hello World";  
char s2[ ] = "World";  
char *s3 = strstr(s1, s2);
```

```
if (s3 == NULL)  
    printf("Not found.");  
else  
    printf("Found position = %d", s3 - s1);
```



■ Practice:

■ Input dynamic string:

- Declare struct Student:
 - Id: fix-sized 8 chars.
 - Name: variable-sized 50 chars.
 - GPA: float.
- Write function to input a student from keyboard.

Contents



- Dynamic string.
- **Binary file.**
- `main()` arguments.



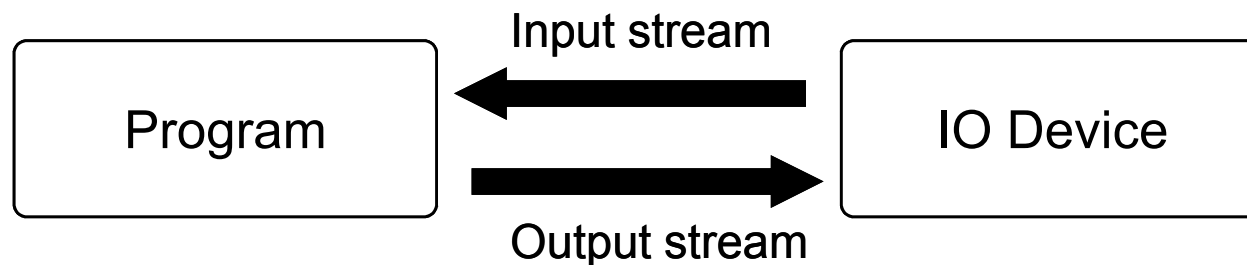
■ IO devices:

- Program is a working machine.
 - ➔ Input data → Program → Output result.
- Where does program retrieve input/output?
 - ➔ From IO devices.
- Types of Devices:
 - Input devices: keyboard, mouse, file, ...
 - Output devices: screen, printer, file, ...
 - ➔ File is both input/output devices.

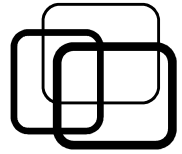


■ IO stream:

- How can program communicate with device?
➔ Through connection called stream.
- Stream: connection between program and device.
- Types of Streams:
 - Input stream: connection from input device to program.
 - Output stream: connection from program to output device.



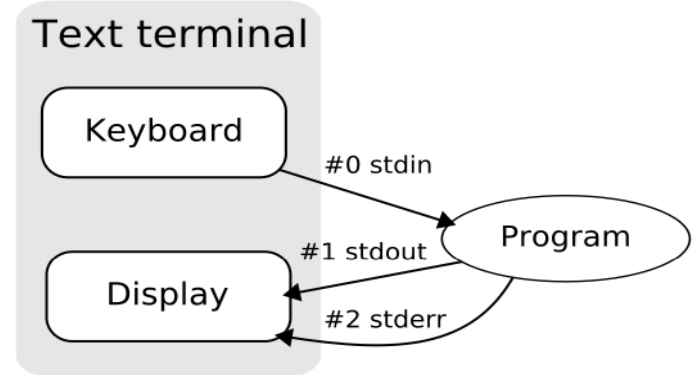
Binary file



■ Stream in C/C++:

■ Built-in streams:

Stream C/C++	Type	Device
stdin / cin	Input stream	Keyboard
stdout / cout	Output stream	Screen
stderr / cerr	Error stream	Screen



■ Usage:

- **fscanf**(<Stream>, "<Input format>", &<Var 1>, ...);
- **fprintf**(<Stream>, "<Output format>", <Var 1>, ...);

`fscanf(stdin, "%d", &x);` // Read from keyboard.

`fprintf(stdout, "Hello World");` // Write to screen.

Binary file



■ File vs. Memory:

Criteria	File	Memory
Processing speed	Slow	Fast
Type of access	Sequential	Random
Cost	Cheap	Expensive
Storage size	Large	Small
Storage time	Persistent	Temporary

■ File vs. Keyboard and Screen:

- User appearance is unnecessary.
- Retrieve input/output repeatedly.
- Communicate with other programs.



■ File stream:

- Connection between program and file.

- Declaration:

- C: **FILE** *<stream> (library <stdio.h>).

- ➔ File pointer.

- C++: **std::fstream** <stream> (library <fstream>).

- Steps to process file:

- Step 1: open file (fopen, freopen).

- Step 2: read/write file (fscanf, fprintf).

- Step 3: close file (fclose).

Binary file



■ Binary mode:

■ Opening mode table:

Mode C/C++	Description
r / std::ios::in	R ead-only, open to read (text).
w / std::ios::out	W rite-only, open to write (text).
a / std::ios::app	A ppend-only, open to append (text).
[r/w/a]+ / use (OR bit)	Combine read/write.
[r/w/a]b / use (OR bit) std::ios::bin	r/w/a in binary mode.

- Allow access raw bytes from file.
- Read/write bytes from file into memory.



■ fread:

■ Read blocks of bytes from file into memory.

- Syntax: **fread**(<memory pointer>, <block size> , <number of blocks>, <file pointer>);
- Return: number of read blocks.
- ➔ End of file: number of read blocks < number of blocks.

```
int    x;
char   *p = new char[ 100 ];
FILE   *f = fopen("C:\\BaiTap.txt", "rb");

if ( f != NULL )
{
    fread( &x, sizeof(int), 1, f );    // Read 4 bytes into x.
    fread( p, sizeof(char), 100, f );// Read 100 bytes into p.
    fclose( f );
}
```



■ fwrite:

- Write blocks of bytes from memory into file.

- Syntax: **fwrite**(<memory pointer>, <block size> , <number of blocks>, <file pointer>);
- Return: number of written blocks.

```
int    x = 123456;
char   s[ ] = "Hello World";
FILE   *f = fopen("C:\\BaiTap.txt", "wb");

if ( f != NULL )
{
    fwrite( &x, sizeof(int), 1, f );           // Write 4 bytes x to file.
    fwrite( s, sizeof(char), strlen(s), f );   // Write 11 bytes s to file.
    fclose( f );
}
```



■ fseek:

■ Move file pointer.

- Syntax: **fseek**(<file pointer>, <offset>, <origin>);
- <origin>:
 - SEEK_SET (beginning of file).
 - SEEK_CUR (current position).
 - SEEK_END (end of file).
- Only works with opening file.

```
FILE *f = fopen("C:\\BaiTap.txt", "r");
if ( f != NULL )
{
    fseek( f, 2, SEEK_CUR ); // Move forward 2 bytes.
    fclose( f );
}
```

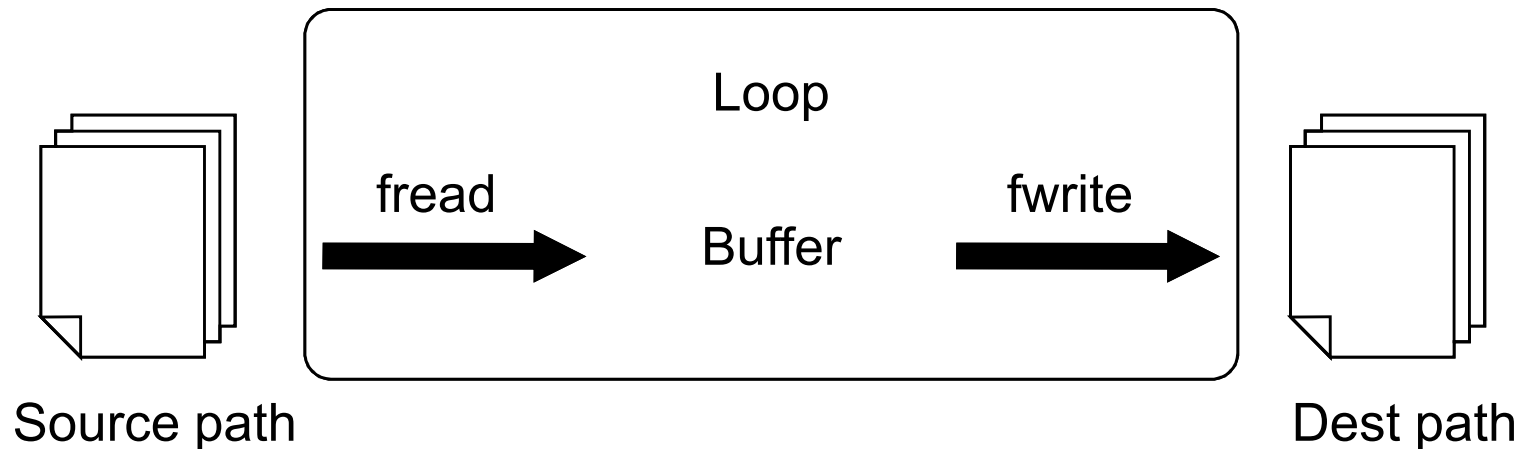
Binary file



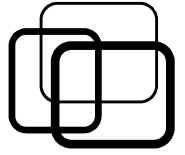
■ Practice:

■ Write function to copy file:

➤ void copyFile(<source path>, <dest path>).

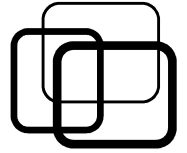


Contents



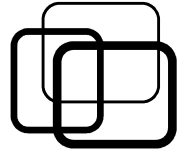
- Dynamic string.
- Binary file.
- **main() arguments.**

main() arguments



- Command-line arguments:
 - Program is a giant function!!
 - How to pass arguments to program?
 - Command-line arguments:
 - Pass arguments to program when calling.
 - main() can get the arguments.
 - Usage:
 - Run program in command-line mode.
 - Syntax: <program> **<arg 1> <arg 2> ...**
C:\>BaiTap\baitap1.exe **hello 5 /abc**
C:\>copy **C:\BaiTap\baitap1.exe D:\Files\baitap1.exe**

main() arguments



■ main() arguments:

■ Declaration:

- Syntax: `int main(int argc, char **argv);`
- `argc`: argument count .
- `argv`: argument variables.
- Arguments passed as strings.
- First argument is program name.

```
int main(int argc, char **argv)
{
    cout << "Number of args = " << argc;
    cout << "Args list:" << endl;
    for (int i = 0; i < argc; i++)
        cout << argv[ i ] << endl;
}
```


Summary



■ File stream:

- Connection between program and device.
- Input/Output in C: printf, scanf.
- Input/Output in C++: cin >>, cout <<.
- File stream: file pointer, fprintf, fscanf.

■ Binary file:

- Read/write raw bytes from file to memory.
- fread, fwrite, fseek.



Summary



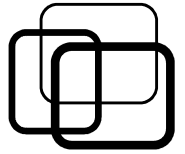
■ Dynamic string:

- Usage dynamic array of chars.
- Do not use “=” to copy string.
- Library <string.h>:
 - strlen, strcpy, strdup.
 - strcmp, strcasecmp, stricmp.
 - strcat, strcpy, strstr.

■ main() arguments:

- Capture command-line arguments.
- Syntax: `int main(int argc, char **argv);`



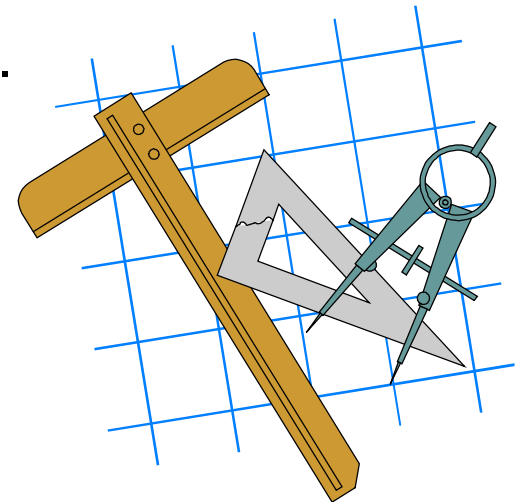


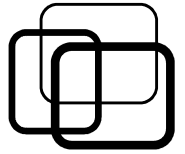
■ Practice 4.1:

Write C/C++ program to do the followings:

- Enter from keyboard a long paragraph until '.' and new line.
- Write to screen:
 - a) Count words in paragraph (alphabets separated by spaces, numbers or punctuations).
 - b) Normalize the paragraph:
 - + Eliminate leading and ending spaces.
 - + Each word separated by only 1 space.
 - + Capitalize first char of each word.

Note: use dynamic string.





■ Practice 4.2:

Write C/C++ program named “COPY” to copy file in command line.

Command-line syntax:

- Syntax 1: copy source file to destination file:

COPY <source file> <destination file>

- Syntax 2: copy source file to destination path (keep filename):

COPY <source file> <destination path>/

- Syntax 3: join file 1 and file 2 to destination file:

COPY <file 1> + <file 2> <destination file>

- Syntax 4: show help:

COPY -?

Note: use dynamic string and binary file.

