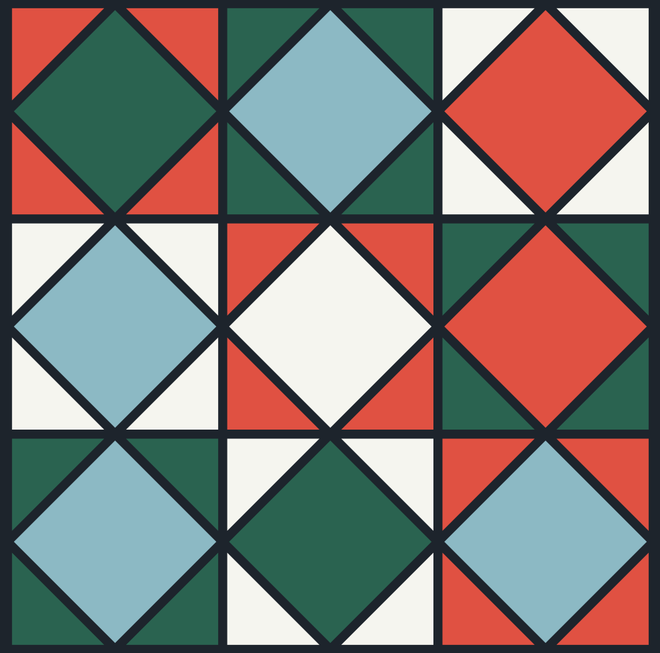
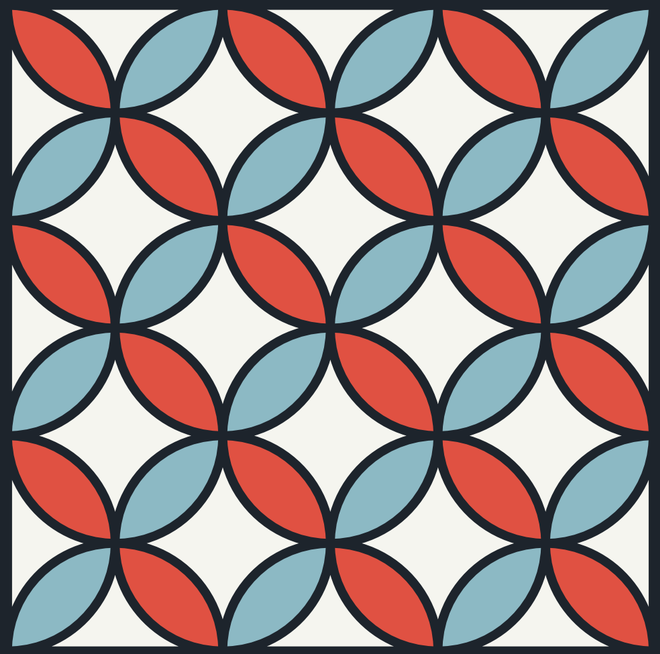
KỸ THUẬT LẬP TRÌNH

CHƯƠNG 9
CÁC ĐỊNH DẠNG ĐẦU VÀO/ĐẦU RA

Thành viên nhóm:
Mạc Anh Kiệt
Nguyễn Văn Nghiêm





OUTLINES

- GIỚI THIỆU
- STREAMS
- ĐỊNH DẠNG ĐẦU RA VỚI HÀM PRINTF
- IN RA SỐ NGUYÊN
- IN RA SỐ THỰC DẤU PHẨY ĐỘNG
- IN RA CHUỖI VÀ KÍ TỰ
- CÁC CHỈ SỐ CHUYỂN ĐỔI KHÁC
- FIELD WIDTH VÀ PRECISION
- SỬ DỤNG FLAGS TRONG HÀM PRINTF
- IN LITERALS VÀ CHUỖI THOÁT
- ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

MỤC TIÊU

01

Hiểu được các luồng nhập và xuất.

02

Có thể sử dụng tất cả các định dạng in phù hợp.

03

Có thể sử dụng tất cả các định dạng nhập phù hợp.

GIỚI THIỆU

- TRÌNH BÀY KẾT QUẢ.
- SCANF VÀ PRINTF.
- LUỒNG (NHẬP VÀ XUẤT).

LUỒNG (STREAM)

CHUỖI KÝ TỰ ĐƯỢC TỔ CHỨC THÀNH DÒNG

- Mỗi dòng bao gồm không hoặc nhiều ký tự và kết thúc bằng ký tự dòng mới.
- ANSI C phải hỗ trợ các dòng ít nhất 254 kí tự.

THỰC HIỆN TẤT CẢ HOẠT ĐỘNG ĐẦU VÀO VÀ ĐẦU RA

THƯỜNG ĐƯỢC CHUYỂN HƯỚNG

- Đầu vào tiêu chuẩn – Bàn phím
- Đầu ra tiêu chuẩn – Màn hình
- Lỗi tiêu chuẩn – Màn hình

ĐỊNH DẠNG ĐẦU RA VỚI HÀM PRINTF

ĐỊNH DẠNG ĐẦU RA VỚI HÀM PRINTF

PRINTF

Định dạng đầu ra chính xác:

- Các thông số chuyển đổi: flags, field, precision, etc.

Có thể thực hiện làm tròn, căn chỉnh cột, căn lề trái phải, chèn các kí tự, định dạng hàm mũ, định dạng thập lục phân, chiều rộng và độ chính xác cố định.

ĐỊNH DẠNG

Printf (format-control-string, other-arguments).
Định dạng điều khiển của chuỗi: mô tả định dạng đầu ra.
Đối số khác: Tương ứng với từng các thông số chuyển đổi trong các định dạng điều khiển của chuỗi.

- Mỗi thông số chuyển đổi bắt đầu với kí hiệu phần trăm, kết thúc bằng mã chuyển đổi.

IN RA SỐ NGUYÊN

Conversion Specifier	Mô tả
d	Hiển thị số nguyên thập phân có dấu.
i	Hiển thị số nguyên thập phân có dấu. (Note: i and d sẽ có cách dùng khác nhau khi dùng với hàm scanf.)
o	Hiển thị số nguyên bát phân không dấu.
u	Hiển thị số nguyên thập phân không dấu.
x hoặc X	Hiển thị một số nguyên thập lục phân không dấu. X biểu diễn các kí tự từ 0–9 và chữ cái từ A–F và x biểu diễn các kí tự 0–9 và chữ cái a–f.
h hoặc l	Đặt trước bất kì thông số chuyển đổi nào để cho biết kiểu số nguyên short hoặclong được hiển thị tương ứng. Chữ cái h và l được gọi chính xác hơn là length modifiers (điều chỉnh về độ dài).

IN RA SỐ NGUYÊN

```c
/* Fig 9.2: fig09_02.c */
/* Using the integer conversion specifiers */
#include <stdio.h>

int main()
{
   printf( "%d\n", 455 );
   printf( "%i\n", 455 );  /* i same as d in printf */
   printf( "%d\n", +455 );
   printf( "%d\n", -455 );
   printf( "%hd\n", 32000 );
   printf( "%ld\n", 2000000000 );
   printf( "%o\n", 455 );
   printf( "%u\n", 455 );
   printf( "%u\n", -455 );
   printf( "%x\n", 455 );
   printf( "%X\n", 455 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
455
455
455
-455
32000
2000000000
707
455
4294966841
1c7
1C7
```

IN RA SỐ THỰC DẤU PHẨY ĐỘNG

In ra số thực dấu phẩy động

- Có dấu thập phân(33.5).
- Exponential notation (Phiên bản máy tính của kí hiệu khoa học (Scientific notation), một cách thể hiện một số quá lớn hoặc quá nhỏ).
  - 150.3 is 1.503 x 10^2 theo kí hiệu khoa học.
  - 150.3 is 1.503E+02 kí hiệu khoa học phiên bản máy tính (E tượng trưng cho kí hiệu mũ).
- e (or E) - in ra số thực theo kí hiệu khoa học phiên bản máy tính.
- f – in dấu phẩy động với ít nhất một chữ số ở bên trái của số thập phân.
- g (or G) - in dạng f or e với không có số không nào ở cuối (1.2300 becomes 1.23).
  - Sử dụng hàm mũ nếu số mũ nhỏ hơn -4 hoặc lớn hơn hoặc bằng độ chính xác (6 kí tự theo mặc định).

IN RA SỐ THỰC DẤU PHẨY ĐỘNG

Conversion Specifier	Mô tả
e hoặc E	Hiển thị giá trị dấu phẩy động trong ký hiệu khoa học.
f	Hiển thị giá trị dấu phẩy động
g hoặc G	Hiển thị giá trị dấu phẩy động ở dạng f or hoặc dạng mũ e (or E).
L	Đặt trước bất kỳ thông số chuyển đổi dấu phẩy động nào để cho biết rằng giá trị dấu phẩy động kép dài được hiển thị.

IN RA SỐ THỰC DẤU PHẨY ĐỘNG

```c
/* Fig 9.4: fig09_04.c */
/* Printing floating-point numbers with
   floating-point conversion specifiers */

#include <stdio.h>

int main()
{
   printf( "%e\n", 1234567.89 );
   printf( "%e\n", +1234567.89 );
   printf( "%e\n", -1234567.89 );
   printf( "%E\n", 1234567.89 );
   printf( "%f\n", 1234567.89 );
   printf( "%g\n", 1234567.89 );
   printf( "%G\n", 1234567.89 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

CÁC MÃ CHUYỂN ĐỔI KHÁC

CÁC MÃ CHUYỂN ĐỔI KHÁC

Conversion Specifier	Mô tả
p	Hiển thị giá trị con trỏ
n	Lưu số kí tự được in ra trong câu lệnh printf Một con trỏ tới một số nguyên được bổ sung làm đối số tương ứng
%	Hiển thị kí hiệu %

CÁC MÃ CHUYỂN ĐỔI KHÁC

```c
/* Fig 9.7: fig09_07.c */
/* Using the p, n, and % conversion specifiers */
#include <stdio.h>

int main()
{
   int *ptr;      /* define pointer to int */
   int x = 12345; /* initialize int x */
   int y;         /* define int y */

   ptr = &x;      /* assign address of x to ptr */
   printf( "The value of ptr is %p\n", ptr );
   printf( "The address of x is %p\n\n", &x );

   printf( "Total characters printed on this line:%n", &y );
   printf( " %d\n\n", y );

   y = printf( "This line has 28 characters\n" );
   printf( "%d characters were printed\n\n", y );

   printf( "Printing a %% in a format control string\n" );

   return 0; /* indicates successful termination */

} /* end main */
```

```
The value of ptr is 0012FF78
The address of x is 0012FF78

Total characters printed on this line: 38

This line has 28 characters
28 characters were printed

Printing a % in a format control string
```

FIELD WIDTH VÀ PRECISION

Field width và precision

- Field width
  - Kích thước của trường được in ra
    - Nếu field width lớn hơn dữ liệu, mặc định được căn lề phải
    - Nếu field width nhỏ hơn dữ liệu, tự động tăng lên phù hợp với dữ liệu
    - Dấu "-" sử dụng 1 vị trí kí tự trong trường
  - Field width của một số nguyên được chèn giữa % và mã chuyển đổi
    - %4d - Field width là 4

Field width và precision

- Precision
  - Ý nghĩa khác nhau phụ thuộc vào kiểu dữ liệu
  - Số nguyên (mặc định là 1)
    - Số chữ số tối thiểu để in
      - Nếu dữ liệu là quá nhỏ, có tiền tố là số 0
  - Số thực dấu phẩy động
    - Số chữ số xuất hiện sau số thập phân (e và f)
      - Với g - Số chữ số có nghĩa lớn nhất
  - Chuỗi
    - Số kí tự tối đa được viết từ chuỗi

Field width và precision

- Precision
  - Định dạng
    - %width.precision : %5.3f
  - Width âm - Căn lề trái >< Width dương - Căn lề phải
  - Precision phải dương
  - Có thể sử dụng các biểu thức số nguyên để xác định giá trị width và precision
  - Đặt dấu * thay cho width hoặc precision
    - Khớp với một đối số int trong danh sách các đối số
    - printf("%*.*f", 7, 2, 56.2356);

FIELD WIDTH VÀ PRECISION

```c
/* Fig 9.8: fig09_08.c */
/* Printing integers right-justified */
#include <stdio.h>

int main()
{
   printf( "%4d\n", 1 );
   printf( "%4d\n", 12 );
   printf( "%4d\n", 123 );
    printf( "%4d\n", 1234 );
    printf( "%4d\n\n", 12345 );

    printf( "%4d\n", -1 );
    printf( "%4d\n", -12 );
    printf( "%4d\n", -123 );
    printf( "%4d\n", -1234 );
    printf( "%4d\n", -12345 );

    return 0; /* indicates successful termination */

} /* end main */
```

```
   1
  12
 123
1234
12345

  -1
 -12
-123
-1234
-12345

--------------------------------
```

FIELD WIDTH VÀ PRECISION

```c
/* Fig 9.9: fig09_09.c */
/* Using precision while printing integers,
   floating-point numbers, and strings */
#include <stdio.h>

int main()
{
   int i = 873;                  /* initialize int i */
   double f = 123.94536;         /* initialize double f */
    char s[] = "Happy Birthday"; /* initialize char array s */

    printf( "Using precision for integers\n" );
    printf( "\t%.4d\n\t%.9d\n\n", i, i );

    printf( "Using precision for floating-point numbers\n" );
    printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );

    printf( "Using precision for strings\n" );
    printf( "\t%.11s\n", s );

    return 0; /* indicates successful termination */

} /* end main */
```

```
 Using precision for integers
        0873
        000000873

Using precision for floating-point numbers
        123.945
        1.239e+002
        124

Using precision for strings
        Happy Birth
```

SỬ DỤNG FLAG TRONG HÀM PRINTF

Sử dụng Flag trong hàm printf

- Flags
  - Bổ sung khả năng định dạng
  - Đặt Flags ngay sau %
  - Một số Flags có thể kết hợp

Flag	Description
-	Căn trái dữ liệu đầu ra
+	Hiện thị dấu + trước giá trị dương và dấu - trước giá trị âm
space	In dấu cách trước giá trị dương
#	Tiền tố 0 cho giá trị đầu ra với mã chuyển đổi bát phân o Tiền tố 0x hoặc 0X cho giá trị đầu ra với mã chuyển đổi thập lục phân x hoặc X Buộc đặt dấu phẩy cho một số thực dấu phẩy động được in bằng mã chuyển đổi e, E, f, g, G không chứa phần phân số. Đôi với mã g và G, các số 0 ở cuối không bị loại bỏ
0	Chèn một trường bằng các số 0 ở đầu

SỬ DỤNG FLAG TRONG HÀM PRINTF

```c
#include <stdio.h>

int main(){
    printf("Example for minus sign\n");
    printf("%10s%10d%10c%10f\n", "MI-TN K65", 22, 'T', 22.3);
    printf("%-10s%-10d%-10c%-10f\n", "MI-TN K65", 22, 'T', 22.3);

    printf("\nExample for plus sign\n");
    printf("%d\t%d\n", 786, -786);
    printf("%+d\t%+d\n", 786, -786);

    printf("\nExample for space flag\n");
    printf("% d\n% d\n", 223, -223);

    int c = 312;
    double p = 312.0;
    printf("\nExample for # flag\n");
    printf("%#o\n", c);
    printf("%#x \t %#X \n", c, c);
    printf("\n%g \t %#g\n", p, p);

    printf("\nExample for 0 flag\n");
    printf("%05d \t %+05d \n", c, c);
}
```

```
Example for minus sign
 MI-TN K65        22         T 22.300000
MI-TN K65 22         T         22.300000

Example for plus sign
786     -786
+786    -786

Example for space flag
 223
-223

Example for # flag
0470
0x138    0X138

312      312.000

Example for 0 flag
00312    +0312

--------------------------------
Process exited after 0.02819 seconds with
Press any key to continue . . .
```

IN RA GIÁ TRỊ CỐ ĐỊNH VÀ CHUỖI THOÁT

LITERALS AND ESCAPE SEQUENCES

GIÁ TRỊ CỐ ĐỊNH (LITERALS)

- Hầu hết các kí tự điều có thể in được.
- Một số ký tự "sự cố", chẳng hạn như dấu ngoặc kép ".
- Phải được thể hiện bằng chuỗi thoát.

- Được thể hiện bằng dấu gạch chéo ngược \ theo sau là kí tự thoát.

Escape Sequences

Escape Sequences	**Description**
\'	Xuất ký tự dấu nháy đơn (').
\"	Xuất ra ký tự dấu ngoặc kép (").
\?	Xuất ra ký tự dấu chấm hỏi (?).
\\	Xuất ra ký tự dấu gạch chéo ngược (\).
\a	Gây ra cảnh báo bằng âm thanh (chuông) hoặc hình ảnh.
\b	Di chuyển con trỏ trở lại một vị trí trên dòng hiện tại.
\f	Di chuyển con trỏ đến đầu trang logic tiếp theo.
\n	Di chuyển con trỏ đến đầu dòng tiếp theo.
\r	Di chuyển con trỏ đến đầu dòng hiện tại.
\t	Di chuyển con trỏ đến vị trí tab ngang tiếp theo.
\v	Di chuyển con trỏ đến vị trí tab dọc tiếp theo.

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

Conversion Specifier	Mô tả
Số nguyên	
d	Đọc một số nguyên thập phân có dấu tùy chọn. Đối số tương ứng là một con trỏ tới số nguyên.
i	Đọc số nguyên thập phân, bát phân hoặc thập lục phân có dấu tùy chọn. Đối số tương ứng là một con trỏ tới số nguyên.
o	Đọc một số nguyên bát phân. Đối số tương ứng là một con trỏ đến số nguyên không dấu.
u	Đọc một số nguyên thập phân không dấu. Đối số tương ứng là một con trỏ đến số nguyên không dấu
x hoặc X	Đọc một số nguyên thập lục phân. Đối số tương ứng là một con trỏ đến số nguyên không dấu.
h hoặc l	Đặt trước bất kỳ từ chỉ định chuyển đổi số nguyên nào để cho biết rằng một số nguyên ngắn hoặc dài sẽ được nhập vào.

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

Conversion Specifier	Mô tả
Số thực dấu phẩy động	
e, E, f, g or G	Đọc giá trị dấu phẩy động. Đối số tương ứng là một con trỏ đến một biến dấu phẩy động.
l or L	Đặt trước bất kỳ thông số kỹ thuật chuyển đổi dấu phẩy động nào để chỉ ra rằng giá trị double hay long double sẽ được nhập vào.
Kí tự và chuỗi	
C	Đọc một ký tự. Đối số tương ứng là một con trỏ tới char, không có null ('\ 0') được thêm vào.
S	Đọc một chuỗi. Đối số tương ứng là một con trỏ tới một mảng kiểu char đủ lớn để chứa chuỗi và một ký tự null ('\ 0') kết thúc — được thêm tự động.
Scan set	
Scan characters	Quét một chuỗi để tìm một tập hợp các ký tự được lưu trữ trong một mảng.
Điều khoản khác	
P	Đọc một địa chỉ có cùng dạng được tạo ra khi một địa chỉ được xuất ra với %p trong câu lệnh printf.
N	Lưu trữ số lượng ký tự đầu vào cho đến nay trong scanf này. Đối số tương ứng là một con trỏ tới số nguyên
%	Bỏ qua dấu phần trăm (%) trong đầu vào.

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

SCANF

- Định dạng đầu vào.
- Sự tương thích:
  - Nhập vào được mọi loại dữ liệu.
  - Nhập các kí tự cụ thể.
  - Bỏ qua các kí tự cụ thể.

ĐỊNH DẠNG

- Scanf (format-control-string, other-arguments);
  - Format-control-string.
  - Miêu tả định dạng của đầu vào.
- Các đối số khác
  - Con trỏ đến các biến nơi đầu vào sẽ được lưu trữ.
  - Có thể bao gồm độ rộng trường để đọc một số ký tự cụ thể từ streams.

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

- Scan tập hợp
  - Tập hợp các ký tự được đặt trong dấu ngoặc vuông[]
    - Bắt đầu bằng %
    - Scan các streams đầu vào, chỉ tìm kiếm các ký tự trong tập hợp scan
    - Khi xảy ra một sự tương hợp, lưu trữ kí tự trong mảng chỉ định
    - Dừng scan khi một kí tự không trong tập scan được tìm thấy
  - Tập scan ngược
    - Dùng dấu mũ ^: [^aeiou]
    - Khiến cho các kí tự không trong tập scan được lưu lại

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

- Bỏ qua các kí tự
  - Bao gồm các kí tự được bỏ qua trong việc kiểm soát định dạng
  - Hoặc, sử dụng * (ký tự loại bỏ phép gán)
    - Bỏ qua bất kỳ loại ký tự nào mà không cần lưu trữ

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

```c
/* Fig 9.18: fig09_18.c */
/* Reading integers */
#include <stdio.h>

int main()
{
   int a; /* define a */
   int b; /* define b */
   int c; /* define c */
    int d; /* define d */
    int e; /* define e */
    int f; /* define f */
    int g; /* define g */

    printf( "Enter seven integers: " );
    scanf( "%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g );

    printf( "The input displayed as decimal integers is:\n" );
    printf( "%d %d %d %d %d %d %d\n", a, b, c, d, e, f, g );

    return 0; /* indicates successful termination */

} /* end main */
```

```
Enter seven integers: -70 -70 070 0x70 70 70 70
The input displayed as decimal integers is:
-70 -70 56 112 56 70 112
```

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

```c
/* Fig 9.20: fig09_20.c */
/* Reading characters and strings */
#include <stdio.h>

int main()
{
   char x;       /* define x */
   char y[ 9 ]; /* define array y */

    printf( "Enter a string: " );
    scanf( "%c%s", &x, y );

    printf( "The input was:\n" );
    printf( "the character \"%c\" ", x );
    printf( "and the string \"%s\"\n", y );

    return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a string: Sunday
The input was:
the character "S" and the string "unday"
```

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

```c
/* Fig 9.21: fig09_21.c */
/* Using a scan set */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   char z[ 9 ]; /* define array z */

   printf( "Enter string: " );
   scanf( "%[aeiou]", z ); /* search for set of characters */

   printf( "The input was \"%s\"\n", z );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter string: ooeeooahah
The input was "ooeeooa"
```

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

```c
/* Fig 9.22: fig09_22.c */
/* Using an inverted scan set */
#include <stdio.h>

int main()
{
   char z[ 9 ] = { '\0' }; /* initialize array z */

   printf( "Enter a string: " );
   scanf( "%[^aeiou]", z ); /* inverted scan set */

   printf( "The input was \"%s\"\n", z );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a string: String
The input was "Str"
```

ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

```c
/* Fig 9.23: fig09_23.c */
/* inputting data with a field width */
#include <stdio.h>

int main()
{
   int x; /* define x */
   int y; /* define y */

   printf( "Enter a six digit integer: " );
   scanf( "%2d%d", &x, &y );

   printf( "The integers input were %d and %d\n", x, y );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a six digit integer: 123456
The integers input were 12 and 3456
```

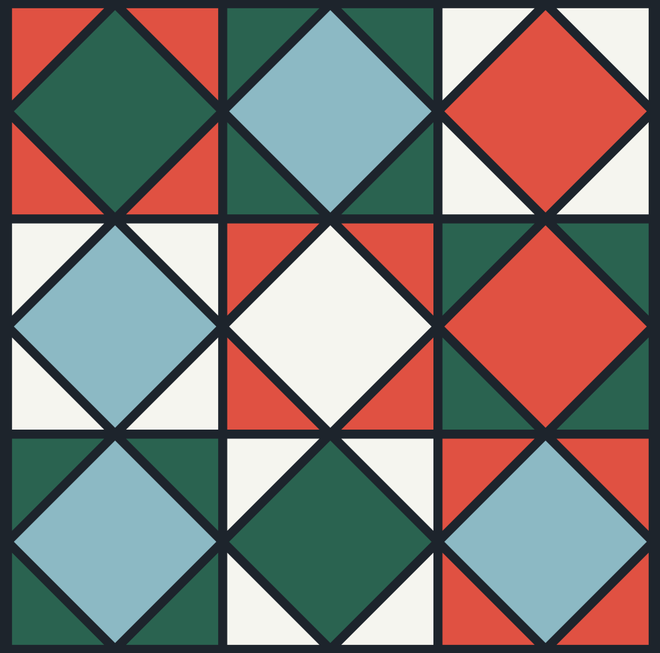
ĐỊNH DẠNG ĐẦU VÀO VỚI HÀM SCANF

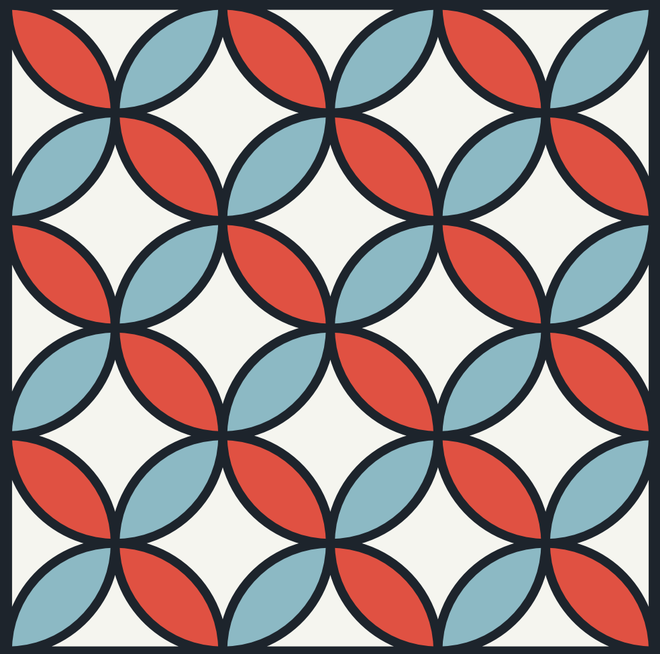
```c
/* Fig 9.24: fig09_24.c */
/* Reading and discarding characters from the input stream */
#include <stdio.h>

int main()
{
   int month1; /* define month1 */
   int day1;   /* define day1 */
   int year1;  /* define year1 */
   int month2; /* define month2 */
   int day2;   /* define day2 */
   int year2;  /* define year2 */

   printf( "Enter a date in the form mm-dd-yyyy: " );
   scanf( "%d%*c%d%*c%d", &month1, &day1, &year1 );

   printf( "month = %d  day = %d  year = %d\n\n", month1, day1, year1 );

   printf( "Enter a date in the form mm/dd/yyyy: " );
   scanf( "%d%*c%d%*c%d", &month2, &day2, &year2 );

   printf( "month = %d  day = %d  year = %d\n", month2, day2, year2 );

   return 0; /* indicates successful termination */

} /* end main */
```

```
Enter a date in the form mm-dd-yyyy: 11-18-2003
month = 11  day = 18  year = 2003

Enter a date in the form mm/dd/yyyy: 11/18/2003
month = 11  day = 18  year = 2003
```

KỸ THUẬT LẬP TRÌNH

CHƯƠNG 10
C STRUCT, UNION, ENUM VÀ THAO TÁC BIT





OUTLINES

- GIỚI THIỆU
- ĐỊNH NGHĨA STRUCT
- KHỞI TẠO STRUCT
- TRUY CẬP ĐẾN CÁC THÀNH VIÊN CỦA STRUCT
- SỬ DỤNG STRUCT VỚI FUNCTION
- TYPEDEF
- VÍ DỤ: HIGH-PERFORMANCE CARD SHUFFLING AND DEADLING SIMULATION
- UNION
- THAO TÁC BIT
- BIT FIELD
- ENUM

MỤC TIÊU

01

Tạo và sử dụng được Struct, union và enum

02

Truyền tham trị và tham chiếu struct tới function

03

Thao tác dữ liệu với toán tử Bit

04

Tạo các trường bit để lưu trữ dữ liệu một cách gọn nhẹ.

GIỚI THIỆU

Giới thiệu

- Struct
  - Tập hợp các biến, các mảng dưới 1 tên
    - Các biến có thể có kiểu dữ liệu khác nhau
  - Thường được sử dụng để xác định các bản ghi sẽ được lưu trong các tệp
  - Kết hợp với con trỏ có thể tạo cấu trúc dữ liệu danh sách liên kết, ngăn xếp, hàng đợi và cây

ĐỊNH NGHĨA STRUCT

Định nghĩa Struct

- Ví dụ:

```
struct card {
    char *face;
    char *suit;
};
```

- struct định nghĩa cấu trúc card
- card là tên cấu trúc và được sử dụng để khai báo các biến của kiểu cấu trúc
- card chứa 2 thành viên kiểu char*
    - face và suit

Định nghĩa Struct

STRUCT

- Có thể chứa một thành viên là một con trỏ đến cùng một kiểu cấu trúc
- Định nghĩa cấu trúc không chiếm bộ nhớ
  - Tạo một kiểu dữ liệu mới được sử dụng để xác định các biến cấu trúc

ĐỊNH NGHĨA

- Giống như các loại biến khác
  - card oneCard, deck[52], *cPtr;
- Có thể thực hiện đồng thời khi định nghĩa struct

```
struct card {
        char *face;
        char *suit;
} oneCard, deck[ 52 ], *cPtr;
```

Định nghĩa Struct

- Các toán tử hợp lệ
  - Gán một struct cho một struct cùng loại
  - Lấy địa chỉ & của 1 struct
  - Truy cập vào các thành viên của struct
  - Sử dụng sizeof để xác định kích thước của 1 struct

KHỞI TẠO STRUCT

Khởi tạo Struct

- Khởi tạo danh sách
  - Ví dụ: card oneCard = {"Three", "Hearts"};
- Câu lệnh gán
  - Ví dụ: card threeHearts = oneCard;
  - Có thể định nghĩa và khởi tạo threeHearts như sau:

```
card threeHearts;
threeHearts.face = "Three";
threeHearts.suit  = "Hearts";
```

TRUY CẬP ĐẾN THÀNH VIÊN STRUCT

Truy cập đến thành viên struct

- Truy cập đến thành viên struct
  - Toán tử dot (.) với các biến của cấu trúc

```
card myCard;
printf("%s", myCard.suit);
```

  - Toán tử -> với các biến con trỏ

```
card *myCardPtr = &myCard;
printf( "%s", myCardPtr->suit);
```

  - myCardPtr->suit tương đương với (*myCardPtr).suit

TRUY CẬP ĐẾN THÀNH VIÊN STRUCT

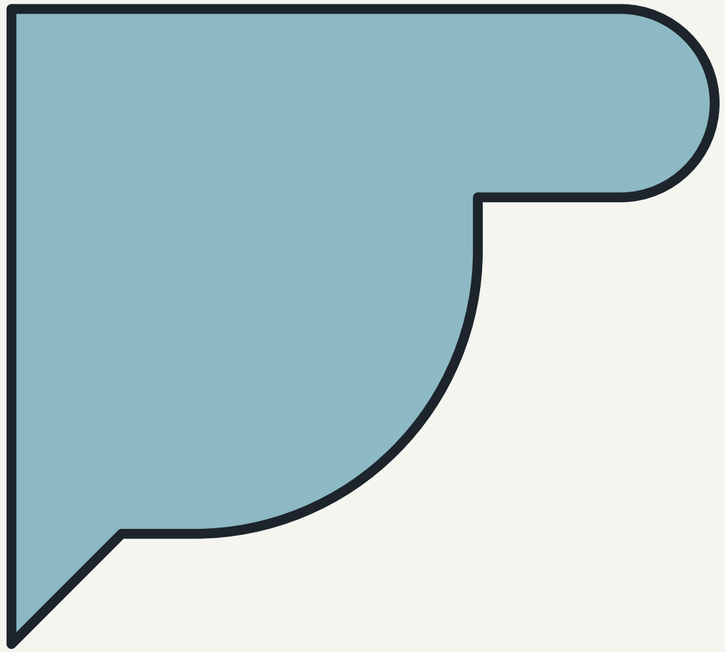
```c
#include <stdio.h>

struct card {   // Dinh nghia struct card
    char *face; // Dinh nghia pointer face
    char *suit; // Dinh nghia pointer suit
};

int main(){
    struct card a;  // Dinh nghia struct a
    struct card *aPtr; // Dinh nghia 1 pointer

    // Dat chuoi vao struct card
    a.face = "Ace";
    a.suit = "Spades";

    aPtr = &a; // gan dia chi cua a cho aPtr

    printf("%s%s%s\n", a.face, " of ", a.suit);
    printf("%s%s%s\n", aPtr->face, " of ", aPtr->suit);
    printf("%s%s%s\n", (*aPtr).face, " of ", (*aPtr).suit);
}

```

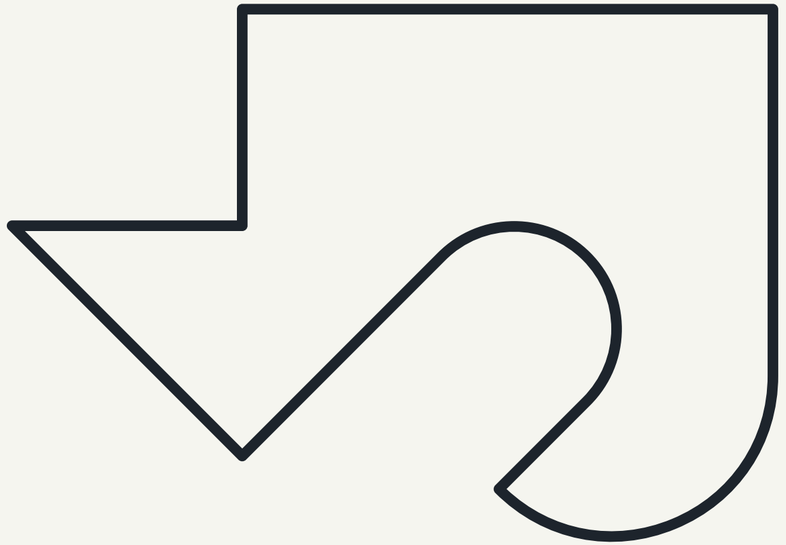
```
Ace of Spades
Ace of Spades
Ace of Spades


--------------------
Process exited after
Press any key to cont
```

SỬ DỤNG STRUCT VỚI FUNCTION

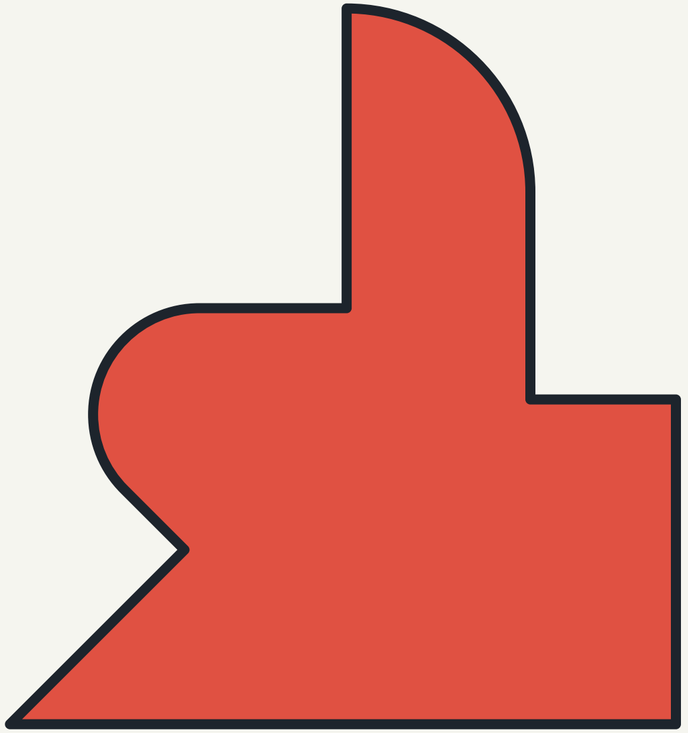
Sử dụng struct với function

- Truyền struct tới function
  - Truyền toàn bộ struct
    - Hoặc truyền các thành viên của struct
  - Truyền struct theo tham chiếu
    - Truyền địa chỉ
    - Truyền tham chiếu đến nó
  - Truyền tham trị của mảng
    - Tạo struct với mảng là thành viên
    - Truyền struct





TYPEDEF

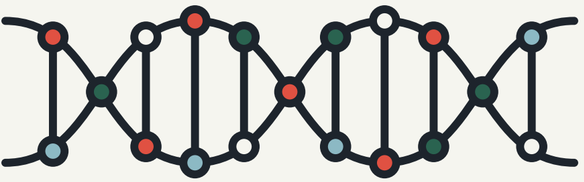


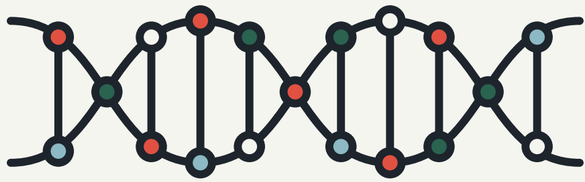
typedef

- Tạo từ đồng nghĩa (bí danh) cho các loại dữ liệu đã được xác định trước đó
- Sử dụng typedef để tạo tên ngắn hơn
- Ví dụ: typedef struct Card *CardPtr;
- Định nghĩa 1 loại tên mới CardPtr như là 1 từ đồng nghĩa của loại struct Card *
- typedef không tạo một kiểu dữ liệu mới
    - Chỉ tạo ra 1 bí danh

VÍ DỤ: HIGH-PERFORMANCE CARD SHUFFLING AND DEALING SIMULATION

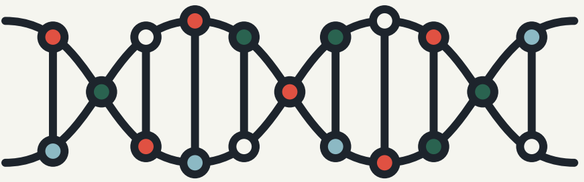
- Mã giả:
  - Tạo 1 mảng struct card
  - Đặt thẻ vào bộ bài
  - Xáo bài
  - Chia bài

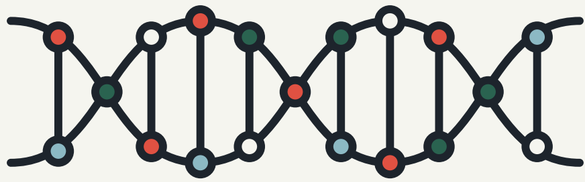




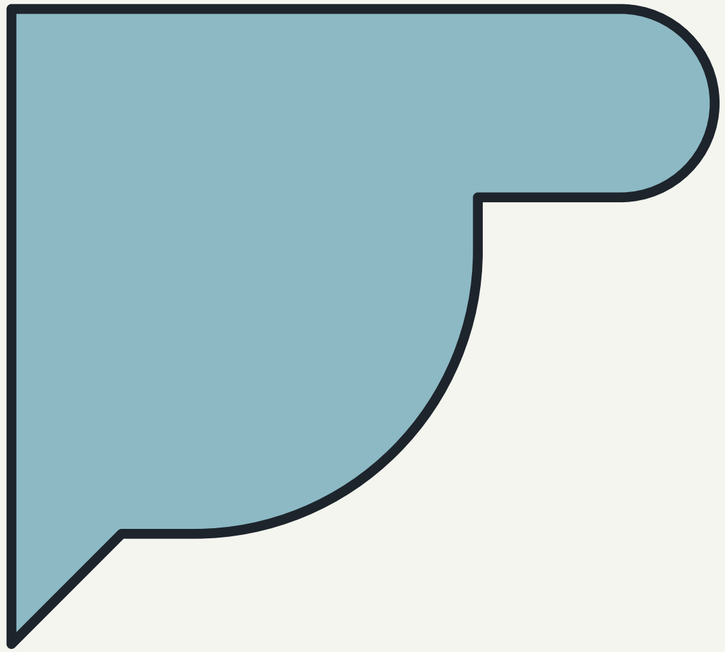
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct card {   // Dinh nghia struct card
    const char *face; // Dinh nghia pointer face
    const char *suit; // Dinh nghia pointer suit
} Card;

// prototypes
void fillDeck(card* const wDeck, const char* wFace[], const char* wSuit[]);
void shuffle(Card* const wDeck);
void deal(const Card* const wDeck);

int main(){
    Card deck[52];  //Dinh nghia mang Card
    // Khoi tao 1 mang pointer
    const char *face[] = {"Ace", "Deuce", "Three", "Four", "Five",
    "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King"};
    const char *suit[] = {"Heart", "Diamonds", "Clubs", "Spades"};

    srand( time(NULL) );

    fillDeck(deck, face, suit); // load the deck with Cards
    shuffle(deck); // put Cards in random order
    deal(deck); // deal all 52 Cards

    return 0;
}
```

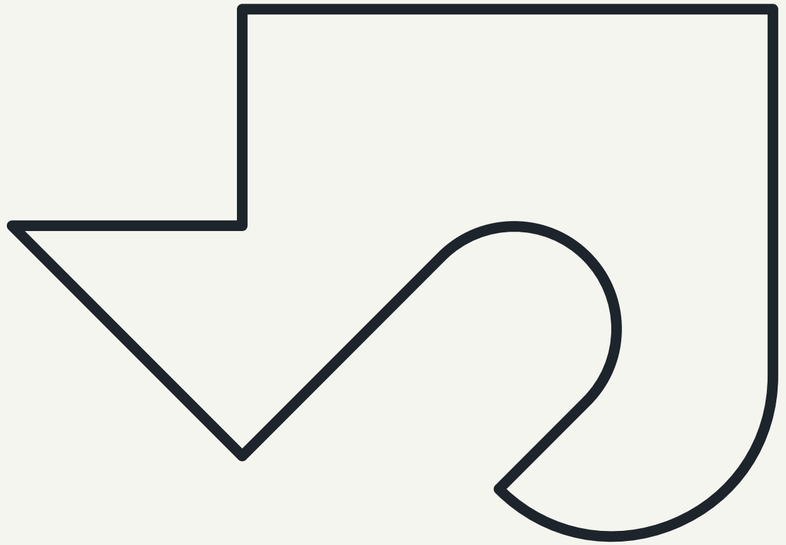
```c
// place strings into card structures
void fillDeck(card* const wDeck, const char* wFace[], const char* wSuit[]){
    int i;
    for(i = 0; i <= 51; i++){
        wDeck[i].face = wFace[i % 13];
        wDeck[i].suit = wSuit[i / 13];
    }
}

// shuffle
void shuffle(Card* const wDeck){
    int i, j;
    Card temp;

    for(i = 0; i <= 51; i++){
        j = rand() % 52;
        temp = wDeck[i];
        wDeck[i] = wDeck[j];
        wDeck[j] = temp;
    }
}

void deal(const Card* const wDeck){
    int i;
    for(i = 0; i <= 51; i++){
        printf("%5s of %-8s%c", wDeck[i].face, wDeck[i].suit,
                                (i + 1) % 2 ? '\t' : '\n');
    }
}
```



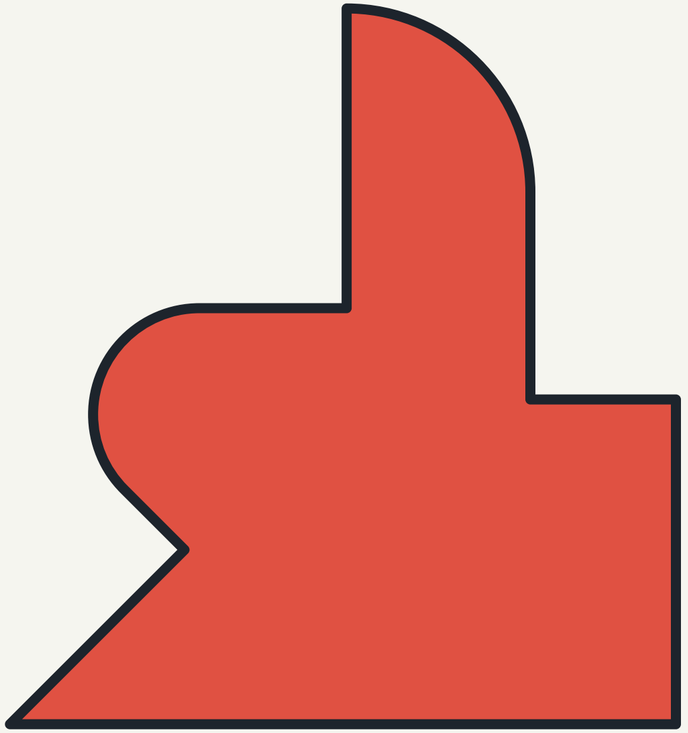


```
Deuce of Clubs            Ace of Clubs
 Five of Diamonds          Ten of Clubs
Seven of Diamonds       Deuce of Diamonds
Three of Spades          Jack of Clubs
Queen of Diamonds       Eight of Spades
Three of Heart          Seven of Spades
Seven of Clubs            Ten of Diamonds
 Nine of Diamonds          Six of Clubs
Three of Clubs            Ace of Heart
  Six of Heart            Ace of Spades
 King of Diamonds          Six of Diamonds
 Five of Clubs          Seven of Heart
 Four of Clubs          Eight of Clubs
Eight of Heart          Deuce of Heart
  Six of Spades          King of Heart
 Five of Heart           Jack of Heart
Deuce of Spades         Queen of Clubs
  Ace of Diamonds        Four of Heart
 Jack of Spades         Eight of Diamonds
 Five of Spades         Queen of Spades
 Nine of Clubs           Nine of Spades
 King of Spades          King of Clubs
 Four of Diamonds        Nine of Heart
Queen of Heart           Jack of Diamonds
 Four of Spades         Three of Diamonds
  Ten of Spades           Ten of Heart


--------------------------------
```





UNIONS



Unions

- Union
  - Giống struct khác ở chỗ các thành viên của union có vùng nhớ chung
  - Bộ nhớ chứa nhiều đối tượng theo thời gian
  - Chỉ chứa một thành viên dữ liệu tại 1 thời điểm
  - Bảo toàn bộ nhớ
  - Chỉ thành viên cuối cùng được xác định mới có thể được truy cập

UNIONS

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

union MITN {
    int girl;
    int boy;
    char name[12];
};

int main(){
    union MITN k65;
    k65.girl = 0;
    k65.boy = 22;
    strcpy(k65.name, "MI TN K65");
    printf("size = %d\ngirl = %d\nboy = %d\nname = %s",
           sizeof(k65), k65.girl, k65.boy, k65.name);
}
```

```
size = 12
girl = 1411402061
boy = 1411402061
name = MI TN K65
--------------------------------
Process exited after 0.02183 secc
Press any key to continue . . .
```

UNIONS

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

union MITN {
    int girl;
    int boy;
    char name[12];
};

int main(){
    union MITN k65;
    printf("size = %d\n", sizeof(k65));
    k65.girl = 0;
    printf("girl = %d\n", k65.girl);
    k65.boy = 22;
    printf("boy = %d\n", k65.boy);
    strcpy(k65.name, "MI TN K65");
    printf("name = %s", k65.name);
}
```

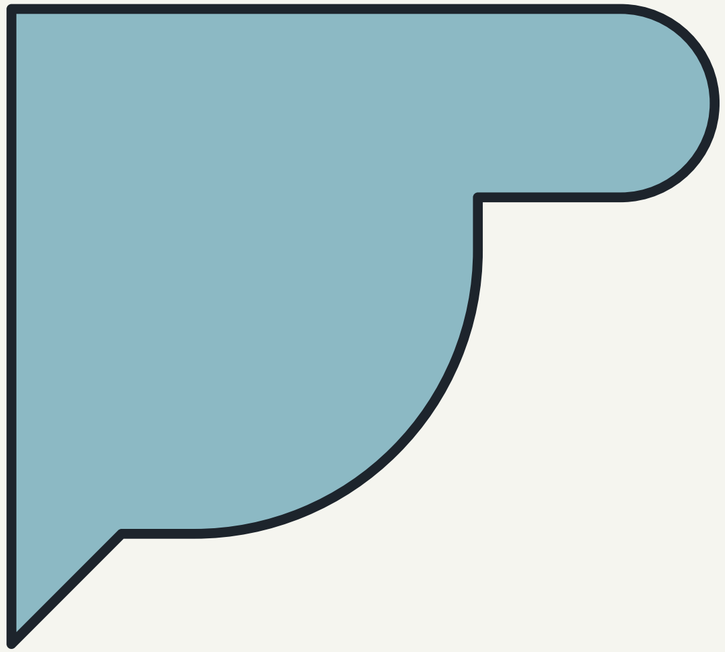
```
size = 12
girl = 0
boy = 22
name = MI TN K65
--------------------------------
Process exited after 0.02499 sec
Press any key to continue . . .
```

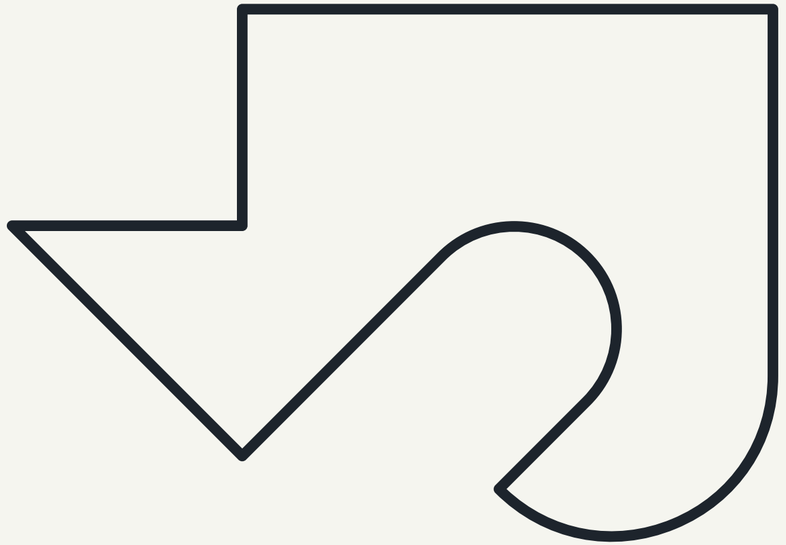
UNIONS

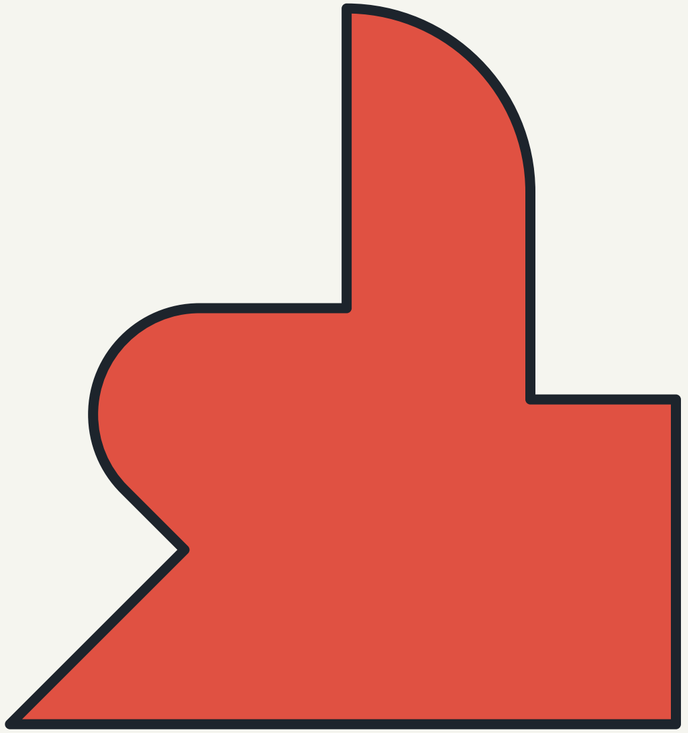
```c
#include <stdio.h>
#include <string.h>

typedef union {
    unsigned short y;     // 2 bytes
    unsigned char ch[2];  // 2 bytes
} val;

int main(){
    val a;
    a.y = 0xa1b2;
    printf("%#x \t %#x", a.ch[0], a.ch[1]);
}
```

```
0xb2     0xa1
--------------------------------
Process exited after 0.02267
Press any key to continue . .
```

THAO TÁC BIT

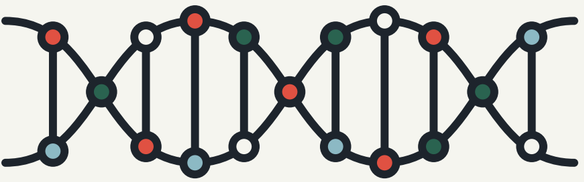




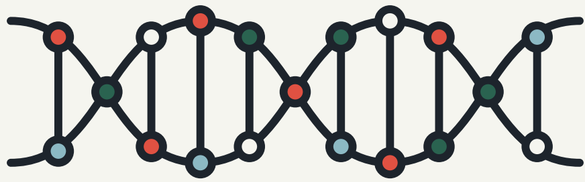


Thao tác BIT

- Tất cả các dữ liệu được biểu diễn dưới dạng chuỗi bit
  - Mỗi bit có thể là 0 hoặc 1
  - Chuỗi 8 bit bằng 1 byte



THAO TÁC BIT



Operator		Description
&	Thao tác BIT AND a & b	Trả về 1 nếu cả 2 hạng tử đều là 1
|	Thao tác OR a | b	Trả về 1 nếu ít nhất 1 trong 2 hạng tử là 1
^	Thao tác XOR a ^ b	Trả về 1 nếu chỉ 1 trong 2 hạng tử là 1
<<	Dịch trái a << 3	Dịch chuyển các bit của toán hạng đầu tiên sang trái bằng số bit được chỉ định bởi toán hạng thứ hai; điền từ bên phải với bit 0
>>	Dịch phải a >> 3	Dịch chuyển các bit của toán hạng đầu tiên sang phải bằng số bit được chỉ định bởi toán hạng thứ hai; phương pháp điền từ bên trái phụ thuộc vào máy
~	Phần bù ~a	Tất cả bit 0 thành 1, tất cả bit 1 thành 0

THAO TÁC BIT

Assignment operators	
&=	a &= b giống a = a & b
|=	a |= b giống a = a &| b
^=	a ^= b giống a = a ^ b
<<=	a <<= 2 giống a = a << 2
>>=	a >>= 3 giống a = a >> 3

THỨ TỰ ƯU TIÊN VÀ SỰ LIÊN KẾT GIỮA CÁC TOÁN TỬ

Operator	Sự liên kết	Loại
() [] . ->	Trái sang phải	Cao nhất
+ - ++ -- ! & * ~ sizeof (type)	Phải sang trái	Một ngôi
* / %	Trái sang phải	Nhân
+ -	Trái sang phải	Thêm vào
<< >>	Trái sang phải	Dịch chuyển
< <= > >=	Trái sang phải	Quan hệ
== !=	Trái sang phải	Sự bằng nhau
&	Trái sang phải	Bitwise AND
^	Trái sang phải	bitwise OR
|	Trái sang phải	bitwise OR
&&	Trái sang phải	logical AND
||	Trái sang phải	logical OR
?:	Phải sang trái	điều kiện
= += -= *= /= &= |= ^= <<= >>= %=	Phải sang trái	Phân công
,	Trái sang phải	Dấu phẩy

TRƯỜNG BIT

Trường BIT

- Trường BIT
  - Thành viên của struct có kích thước (tính bằng bit) đã được chỉ định
  - Cho phép sử dụng bộ nhớ tốt hơn
  - Phải được xác định như là int hoặc unsigned
  - Không thể truy cập các bit riêng lẻ
- Định nghĩa trường BIT
  - Sau unsigned hoặc int với dấu : và 1 hằng số nguyên đại diện cho chiều rộng của trường
  - Ví dụ:

```
struct BitCard {
        unsigned face : 4;
        unsigned suit : 2;
        unsigned color : 1;
};
```

Trường BIT

- Trường BIT không tên
  - Trường được sử dụng làm phần đệm trong cấu trúc
  - Không có gì có thể được lưu trong các BIT
  - Ví dụ:

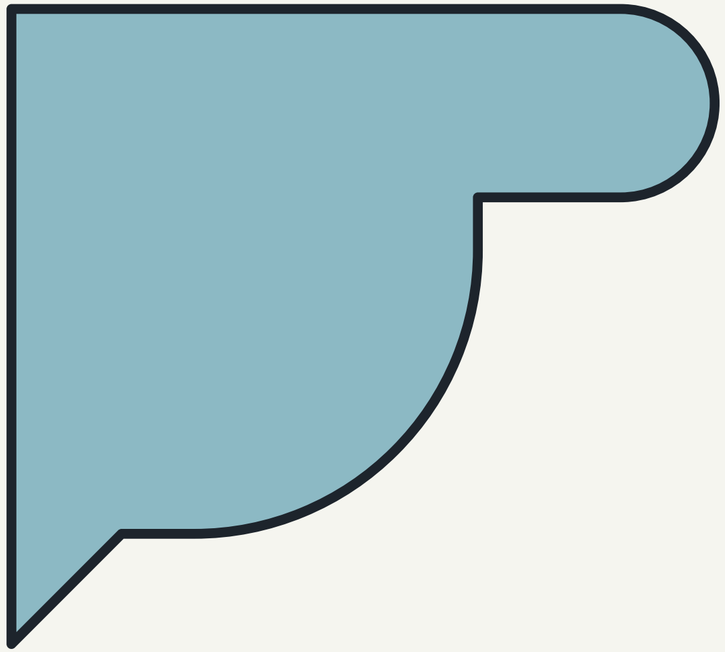
```
struct Example {
        unsigned a : 13;
        unsigned: 3;
        unsigned b : 4;
 };
```

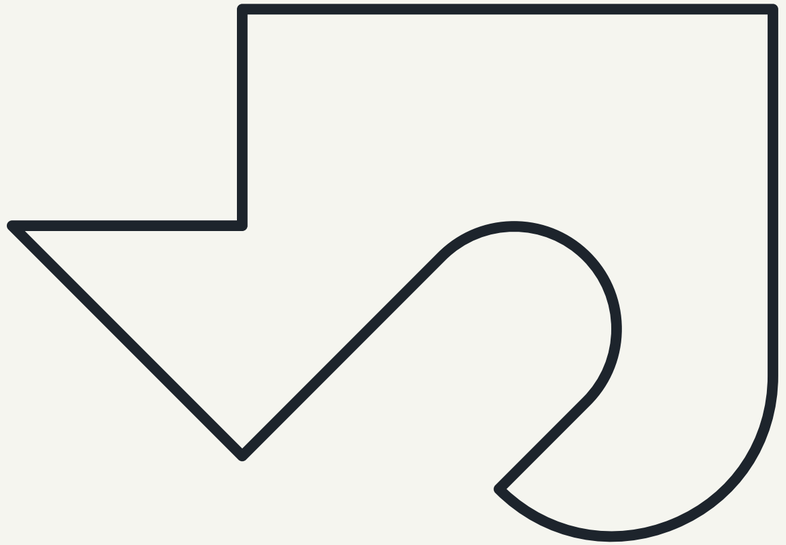
  - Trường bit không tên với độ rộng bằng 0 sẽ căn chỉnh trường bit tiếp theo thành ranh giới đơn vị lưu trữ mới

UNIONS

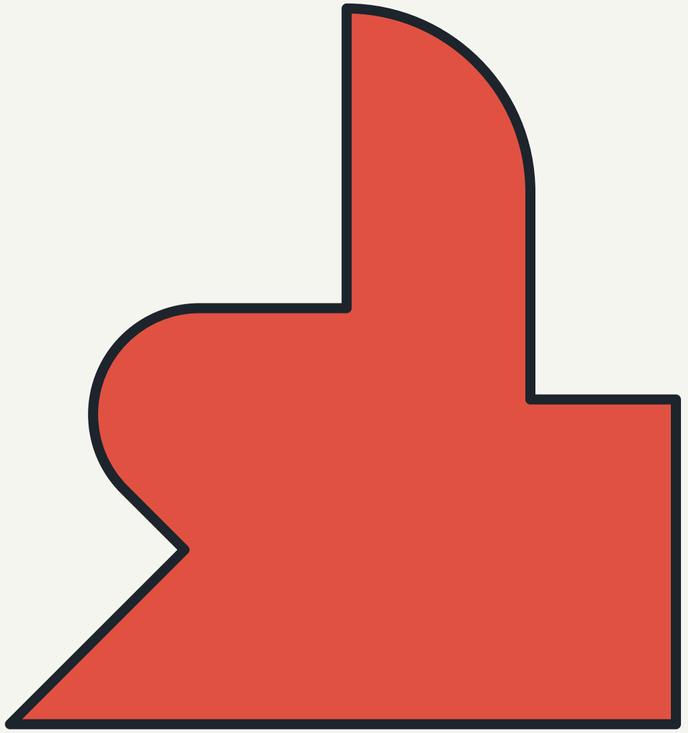
```c
#include <stdio.h>

union{
    struct{
        unsigned short a1;
        unsigned short a2;
    } s;

    struct{
        unsigned n1 : 4;
        unsigned    : 12;
        unsigned n2 : 7;
        unsigned    : 1;
        unsigned n3 : 8;
    } f;
} u;
int main(){
    u.s.a1 = 53257; // 1101 0000 0000 1001
    u.s.a2 = 55545; // 1101 1000 1111 1001

    printf("%d\n", u.f.n1);   // 1001 = 9
    printf("%d\n", u.f.n2);   // 111 1001 = 121
    printf("%d\n", u.f.n3);   // 1101 1000 = 216

}
```

```
9
121
216


--------------------------------
Process exited after 0.02367 s
Press any key to continue . .
```





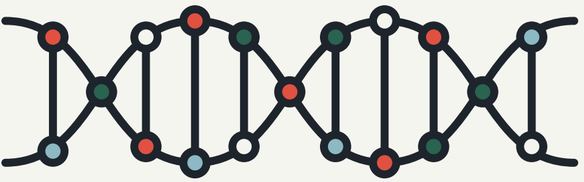
ENUM



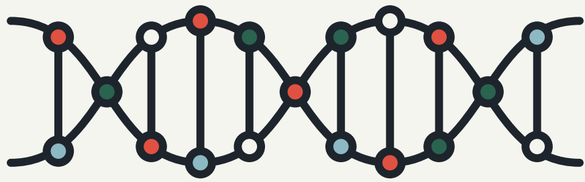
ENUM

- Enumeration
  - Tập hợp các hằng số nguyên được đại diện bởi số nhận dạng
  - Hằng số liệt kê giống như hằng số tượng trưng có giá trị được đặt tự động
  - Giá trị bắt đầu từ 0 và được tăng lên 1
    - Giá trị có thể được đặt với =
    - Cần tên là constant
  - Ví dụ:

    enum Months { JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC};
    - Tạo 1 kiểu enum Months trong đó các số nhận dạng được đặt thành các số nguyên 1 đến 12
  - Các biến liệt kê chỉ có thể giả định các giá trị hằng số liệt kê của chúng (không phải là các biểu diễn số nguyên)



ENUM



```c
/* Fig. 10.18: fig10_18.c
   Using an enumeration type */
#include <stdio.h>

/* enumeration constants represent months of the year */
enum months { JAN = 1, FEB, MAR, APR, MAY, JUN,
              JUL, AUG, SEP, OCT, NOV, DEC };

int main()
{
   enum months month; /* can contain any of the 12 months */

   /* initialize array of pointers */
   const char *monthName[] = { "", "January", "February", "March",
      "April", "May", "June", "July", "August", "September", "October",
      "November", "December" };

   /* loop through months */
   for ( month = JAN; month <= DEC; month++ ) {
      printf( "%2d%11s\n", month, monthName[ month ] );
   } /* end for */

   return 0; /* indicates successful termination */
} /* end main */
```

```
 1    January
 2   February
 3      March
 4      April
 5        May
 6       June
 7       July
 8     August
 9  September
10    October
11   November
12   December
```

THANK YOU!

MỜI MỌI NGƯỜI TIẾP TỤC THAM GIA GIẢI
MỘT VÀI BÀI TẬP CÙNG CHÚNG MÌNH!