Một số kiểu dữ liệu và từ khóa đặc biệt

Nội dung

1. enum
2. struct
3. typedef
4. union
5. define
6. Toán tử điều kiện ‘?’

1. Từ khóa enum

- Định nghĩa một tập hợp các hằng số kiểu int
- Cách viết:

  enum tên_kiểu {tên_hằng = [giá_trị], ...} các_biến;
- tên_kiểu có thể có hoặc không, nó là tên cho tập hợp.
- tên_hằng là tên của một hằng, nó có thể được gán giá_trị.
- các_biến có thể có hoặc không. Chúng có kiểu enum.

- CHÚ Ý: giá_trị phải là một số nguyên. Nếu bị bỏ qua, nó được tính bằng giá trị của hằng số nguyên đứng trước cộng với 1.
- Đối với tên_hằng đầu tiên, giá trị mặc định là 0.

1. Từ khóa enum

Ví dụ:

```
enum modes { LASTMODE = -1, BW40 = 0,
C40, BW80, C80, MONO = 7 };
```

"modes" là tên kiểu.

"LASTMODE", "BW40", "C40", ... là các tên hằng.

Giá trị của C40 là 1, BW80 = 2, C80 = 3, MONO = 7

1. Từ khóa enum

Ví dụ:

- Định nghĩa kiểu tháng có tên là Month gồm các hằng {Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec}
- Viết chương trình hiển thị số ngày trong từng tháng của một năm

1. Từ khóa enum

```
1.  #include<stdio.h>
2.  #include<conio.h>
3.  int main()
4.  {
5.      enum Month {Jan = 0, Feb, Mar, Apr, May, Jun, Jul, Aug,Sep, Oct, Nov, Dec};
6.      char DaysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,31};
7.      enum Month M;
8.      clrscr();
9.      for(M = Jan; M <= Dec; M++)
10.         printf("\nthang %d co so ngay la: %d", M, DaysInMonth[M] );
11.     getch();
12.     return 0;
13. }
```

1. Từ khóa enum

- Một số câu lệnh enum:

  enum tk {pt1, pt2,…} tb1, tb2,…;

  enum tk {pt1, pt2,…};

  enum {pt1, pt2, …} tb1, tb2, …;

  enum {pt1, pt2, …};

- trong đó:
  - tk là tên kiểu enum (một kiểu dữ liệu mới)
  - pt1, pt2,… là tên các phần tử (các hằng)
  - tb1, tb2, …là tên các biến kiểu enum

2. Cấu trúc – struct

Định nghĩa

```
struct Tên_kiểu_cấu_trúc {
    Kiểu1  thanh_phan1, tthanh_phan2 ;
    Kiểu2  tthanh_phan3, thanh_phan4 ;
    ...
} biến_cấu_trúc ;
```

Lưu ý:

- Tên_kiểu_cấu_trúc và biến_cấu_trúc, có thể có hoặc không, nhưng một trong hai phải có.

2. Cấu trúc – struct

Ví dụ 1:

```
struct ngay {
  int ngay_thu;
  char ten_thang[10];
   int nam;
}
```

Ví dụ 2:

```
Struct nhan_cong
    {
    char  ten[15];
    char dia_chi[25];
    double bac_luong;
    struct ngay ngay_sinh;
    }
```

Cấu trúc – Khai báo

- Cách 1:

```
struct ngay {
   int ngay_thu;
   char ten_thang[10];
    int nam;
} ngay_di, ngay_den;
```

Cấu trúc – Khai báo

- Cách 2:

```
struct  {
   int ngay_thu;
   char ten_thang[10];
    int nam;
} ngay_di, ngay_den;
```

Cấu trúc – Khai báo

- Cách 3:

  Struct nhan_cong nguoi_1, nguoi_2;

Ví dụ 2:

```
Struct nhan_cong
    {
    char  ten[15];
    char dia_chi[25];
    double bac_luong;
    struct ngay ngay_sinh;
    }
```

Truy nhập đến các thành phần Cấu trúc

ten_bien_cau_truc.ten_thanh_phan

ten_bien_cau_truc.ten_bien_cau_truc.ten_thanh_phan

Ví dụ:

```
printf (“%s”, nguoi_1.ten);

s= nguoi_1.luong + nguoi_2.luong;

printf(“Nam sinh cua nguoi_1 la %d”, nguoi_1.ngay_sinh.nam);
```

3. Định nghĩa kiểu dữ liệu - typedef

- Cú pháp: **typedef Kiểu_cần_định_nghĩa Tên_kiểu ;**
- Định nghĩa một **Tên_kiểu** thay thế cho **Kiểu_cần_định_nghĩa**.
- Sau khi định nghĩa thì việc khai báo **Tên_kiểu** sẽ tương đương với **Kiểu_cần_định_nghĩa**.

- Ví dụ:

```
typedef  unsigned char  byte;
typedef struct {
      double re, im;
} complex;
```

Hãy so sánh với:

```
unsigned char byte;
struct {
      double re, im;
} complex;
```

4. Từ khóa union

- Một union là giống với một cấu trúc (struct), ngoại trừ nó cho phép bạn định nghĩa các biến mà chúng chung nhau không gian lưu trữ.

- Cách viết:

```
union [tên_kiểu_union] {
  kiểu các_biến;
  ...
} [các_biến_union] ;
```

4. Từ khóa union

- Ví dụ:

```
union int_or_long {
  int     i;
  long    l;
} a_number;
```

- Không giống với một cấu trúc, các biến a_number.i and a_number.l chiếm cùng vị trí bộ nhớ. Do đó, việc ghi một biến sẽ đè lên biến còn lại.

- Các thành phần của một union được truy cập giống như là đối với một cấu trúc.

5. Từ khoá define

- Được dùng để định nghĩa macro:

  #define tên_marco dãy_biểu_thức

- Khi thực thi chương trình máy tính sẽ tự động thay thế tên_marco bằng dãy_biểu_thức.

- Ví dụ:

  ```
  #define MAX 100
  #define  max(a,b) (a)>(b)?(a):(b)
  ```

6. Toán tử điều kiện (? :)

- Toán tử ?: là một toán tử có toán hạng.
- Cách viết:

  điều_kiện ? E1 : E2
- Ý nghĩa:
  - Nếu điều_kiện đúng thì E1 được tính và bỏ quả E2.
  - Nếu điều_kiện sai thì E2 được tính và bỏ quả E1.
  - Giá trị của điều_kiện ? E1 : E2 là giá trị của biểu thức được tính.
- Ví dụ

  max = a > b? a : b ;

Truy nhập đến các thành phần Cấu trúc

- Để truy nhập các thành phần trong một cấu trúc thì cần chỉ rõ nó là thành phần nào và thuộc biến (con trỏ) cấu trúc nào:

- Nếu truy cập thông qua biến cấu trúc thì cần viết là tên_biến.thành_phần

```
strcpy(my_friend.name, "Mr. Wizard");
my_friends[10].age = 83;
```

- Nếu truy cập thông qua con trỏ cấu trúc thì cần viết là tên_con_trỏ ➔ thành_phần

```
strcpy(pmyfriend➔name, "Mr. Wizard");
pmyfriend➔age = 83;
```