# Union trong C

### Khái niệm về union

Trong ngôn ngữ lập trình C, **union** là một kiểu dữ liệu đặc biệt cho phép lưu trữ nhiều kiểu dữ liệu khác nhau tại cùng một vị trí bộ nhớ. Tất cả các thành viên trong union đều dùng chung một vùng nhớ, vì vậy nó chỉ có thể lưu trữ một giá trị tại một thời điểm.

Cú pháp khai báo union:

|  |
| --- |
| union UnionName {      dataType1 member1;      dataType2 member2;      ...  }; |

Ví dụ:

|  |
| --- |
| #include <stdio.h>  union Data {      int i;      float f;      char str[20];  };  int main() {      union Data data;        data.i = 10;      printf("data.i = %d\n", data.i);        data.f = 220.5;      printf("data.f = %.2f\n", data.f);      strcpy(data.str, "Hello");      printf("data.str = %s\n", data.str);        return 0;  } |

### Đặc điểm của union

**Dùng chung bộ nhớ**: Các thành viên của union chia sẻ chung một vùng nhớ, nên khi gán giá trị mới cho một thành viên, giá trị của các thành viên khác sẽ bị thay đổi.

**Kích thước của union**: Kích thước của union bằng kích thước của thành viên có kích thước lớn nhất.

**Truy xuất dữ liệu**: Tại một thời điểm, chỉ nên sử dụng một thành viên của union, nếu không dữ liệu có thể bị ghi đè.

### So sánh union và struct

| 1. **Tiêu chí** | 1. **union** | 1. **struct** |
| --- | --- | --- |
| Bộ nhớ | Chia sẻ cùng một vùng nhớ | Mỗi thành viên có vùng nhớ riêng |
| Kích thước | Bằng kích thước của thành viên lớn nhất | Tổng kích thước của tất cả thành viên |
| Tốc độ truy xuất | Nhanh hơn do dùng chung bộ nhớ | Chậm hơn do có vùng nhớ riêng |
| Sử dụng khi nào? | Khi chỉ cần lưu một giá trị tại một thời điểm | Khi cần lưu đồng thời nhiều giá trị |

Ví dụ về sự khác biệt:

|  |
| --- |
| #include <stdio.h>  #include <string.h>  struct StructExample {      int i;      float f;      char str[20];  };  union UnionExample {      int i;      float f;      char str[20];  };  int main() {      struct StructExample s;      union UnionExample u;      printf("Size of struct: %lu\n", sizeof(s));      printf("Size of union: %lu\n", sizeof(u));      return 0;  } |

**Kết quả:**

|  |
| --- |
| Size of struct: 28  Size of union: 20 |

→ Struct chiếm nhiều bộ nhớ hơn vì mỗi thành viên có vùng nhớ riêng.

### Ứng dụng của union

**Tiết kiệm bộ nhớ** khi chỉ cần sử dụng một thành viên tại một thời điểm.

**Xử lý dữ liệu kiểu khác nhau** tại cùng một vị trí bộ nhớ, chẳng hạn như:

* Biểu diễn dữ liệu dưới dạng nhiều kiểu khác nhau.
* Dùng trong hệ thống nhúng để truy cập vào phần cứng.
* Đọc dữ liệu từ file nhị phân.

Ví dụ thực tế về đọc dữ liệu dưới dạng kiểu khác nhau:

|  |
| --- |
| #include <stdio.h>  union Number {      int i;      float f;  };  int main() {      union Number num;      num.i = 5;      printf("As integer: %d\n", num.i);      num.f = 3.14;      printf("As float: %.2f\n", num.f);      return 0;  } |

**Lưu ý**: Nếu num.i được gán trước, sau đó num.f được gán, thì num.i sẽ bị ghi đè.