1. Basic Union Declaration

```c
#include <stdio.h>

union MyUnion {
    int i;
    float f;
    char c;
};

int main() {
    union MyUnion u;
    u.i = 42;
    printf("Value of i: %d\n", u.i);
    u.f = 3.14;
    printf("Value of f: %f\n", u.f);
    u.c = 'A';
    printf("Value of c: %c\n", u.c);

    return 0;
}
```

2. Union Inside a Struct

```c
#include <stdio.h>

struct Employee {
    char name[30];
    union {
        int employeeID;
        float salary;
    } info;
};
int main() {
    struct Employee emp;
    strcpy(emp.name, "John");
    emp.info.employeeID = 101;
```

```
    printf("Name: %s, Employee ID: %d\n", emp.name, emp.info.employeeID);
    emp.info.salary = 50000.0;
    printf("Name: %s, Salary: %.2f\n", emp.name, emp.info.salary);
    return 0;
}
```

3. Using Union to Save Space

```
#include <stdio.h>
union SpaceSaving {
    int i;
    char c;
};
int main() {
    printf("Size of SpaceSaving: %lu\n", sizeof(union SpaceSaving));
    return 0;
}
```

4. Union as Function Parameter

```
#include <stdio.h>

union MyUnion {
    int i;
    float f;
    char c;
};

void printUnion(union MyUnion u) {
    printf("Value: %d\n", u.i);
}

int main() {
    union MyUnion u;
    u.i = 42;
    printUnion(u);
```

```
    return 0;
}
```

5. Union with Bit Fields

```
#include <stdio.h>
union Flags {
    struct {
        unsigned int flag1 : 1;
        unsigned int flag2 : 1;
        unsigned int flag3 : 1;
    } bits;
    unsigned int all_flags;
};
int main() {
    union Flags flags;
    flags.bits.flag1 = 1;
    flags.bits.flag2 = 0;
    flags.bits.flag3 = 1;

    printf("Flag 1: %d, Flag 2: %d, Flag 3: %d\n", flags.bits.flag1,
flags.bits.flag2, flags.bits.flag3);
    printf("All Flags: %u\n", flags.all_flags);

    return 0;
}
```

6. Union with Enum Inside

```

#include <stdio.h>

enum DataType {
    INT,
    FLOAT,
    CHAR
```

```
};
union MyUnion {
    int i;
    float f;
    char c;
    enum DataType type;
};
int main() {
    union MyUnion u;
    u.type = FLOAT;

    switch (u.type) {
        case INT:
            u.i = 42;
            printf("Value of i: %d\n", u.i);
            break;
        case FLOAT:
            u.f = 3.14;
            printf("Value of f: %f\n", u.f);
            break;
        case CHAR:
            u.c = 'A';
            printf("Value of c: %c\n", u.c);
            break;
    }
    return 0;
}
```

7. Union Array

```
#include <stdio.h>
union Number {
    int integer;
    float floating;
};
int main() {
    union Number numbers[5];
    for (int i = 0; i < 5; i++) {
```

```
        numbers[i].integer = i;
        printf("Integer: %d, Float: %f\n", numbers[i].integer,
numbers[i].floating);
    }
    return 0;
}
```

8. Union with Pointers

```
#include <stdio.h>
union Data {
    int *ip;
    float *fp;
};
int main() {
    union Data data;
    int num = 42;
    data.ip = &num;
    printf("Value: %d\n", *(data.ip));
    float pi = 3.14;
    data.fp = &pi;
    printf("Value: %f\n", *(data.fp));
    return 0;
}
```

9. Union Typedef

```
#include <stdio.h>
typedef union {
    int i;
    float f;
    char c;
} MyUnion;
int main() {
    MyUnion u;
    u.i = 42;
    printf("Value of i: %d\n", u.i);
```

```
    u.f = 3.14;
    printf("Value of f: %f\n", u.f);
    u.c = 'A';
    printf("Value of c: %c\n", u.c);
    return 0;
}
```

10. Union with Function Pointers

```
#include <stdio.h>
union FunctionPointer {
    int (*add)(int, int);
    int (*subtract)(int, int);
};
int add(int a, int b) {
    return a + b;
}
int subtract(int a, int b) {
    return a - b;
}
int main() {
    union FunctionPointer fp;
    fp.add = add;
    printf("Add: %d\n", fp.add(5, 3));
    fp.subtract = subtract;
    printf("Subtract: %d\n", fp.subtract(5, 3));
    return 0;
}
```

11. Union for Color Representation

```
#include <stdio.h>

union Color {
    unsigned int rgba;
    struct {
        unsigned char r;
```

```
        unsigned char g;
        unsigned char b;
        unsigned char a;
    } components;
};

int main() {
    union Color red;
    red.components.r = 255;
    red.components.g = 0;
    red.components.b = 0;
    red.components.a = 255;

    printf("Red: rgba(%u, %u, %u, %u)\n", red.components.r,
red.components.g, red.components.b, red.components.a);

    return 0;
}
```

12. Union for Currency Conversion

```
#include <stdio.h>
union Money {
    float dollars;
    float euros;
    float yen;
};
int main() {
    union Money money;
    money.dollars = 100.0;
    printf("$100 in dollars: $%.2f\n", money.dollars);

    money.euros = money.dollars * 0.85;
    printf("$100 in euros: €%.2f\n", money.euros);
    money.yen = money.dollars * 110.56;
    printf("$100 in yen: ¥%.2f\n", money.yen);
    return 0;
}
```

13. Union for Temperature Conversion

```c
#include <stdio.h>

union Temperature {
    float celsius;
    float fahrenheit;
};

int main() {
    union Temperature temp;
    temp.celsius = 25.0;
    printf("25°C in Celsius: %.2f\n", temp.celsius);

    temp.fahrenheit = (temp.celsius * 9 / 5) + 32;
    printf("25°C in Fahrenheit: %.2f\n", temp.fahrenheit);

    return 0;
}
```

14. Union for Time Representation

```c
#include <stdio.h>

union Time {
    unsigned int total_seconds;
    struct {
        unsigned int hours;
        unsigned int minutes;
        unsigned int seconds;
    } components;
};

int main() {
    union Time t;
    t.components.hours = 2;
```

```
    t.components.minutes = 30;
    t.components.seconds = 45;

    printf("Time: %02d:%02d:%02d\n", t.components.hours,
t.components.minutes, t.components.seconds);

    t.total_seconds = (t.components.hours * 3600) + (t.components.minutes
* 60) + t.components.seconds;
    printf("Total seconds: %u\n", t.total_seconds);

    return 0;
}
```

15. Union for Storing Complex Numbers

```
#include <stdio.h>

struct Complex {
    union {
        float real;
        float imaginary;
    } parts[2];
};

int main() {
    struct Complex c;
    c.parts[0].real = 2.0;
    c.parts[1].imaginary = -3.0;

    printf("Complex number: %f + %fi\n", c.parts[0].real,
c.parts[1].imaginary);

    return 0;
}
```

16. Union for Date Representation

```c
#include <stdio.h>
union Date {
    unsigned int packed_date;
    struct {
        unsigned int day : 5;
        unsigned int month : 4;
        unsigned int year : 12;
    } components;
};
int main() {
    union Date d;
    d.components.day = 25;
    d.components.month = 9;
    d.components.year = 2023;

    printf("Date: %d/%d/%d\n", d.components.day, d.components.month,
d.components.year);
    return 0;
}
```

17. Union for Storage Units Conversion

```c
#include <stdio.h>

union Storage {
    unsigned long long bits;
    struct {
        unsigned long long gigabytes : 10;
        unsigned long long megabytes : 10;
        unsigned long long kilobytes : 10;
        unsigned long long bytes : 34;
    } components;
};

int main() {
    union Storage s;
    s.components.gigabytes = 2;
```

```
    s.components.megabytes = 512;
    s.components.kilobytes = 2048;
    s.components.bytes = 1234567890;

    printf("Storage: %lld GB %lld MB %lld KB %lld B\n",
s.components.gigabytes, s.components.megabytes, s.components.kilobytes,
s.components.bytes);

    return 0;
}
```

18. Union for IP Address Representation

```
#include <stdio.h>

union IPAddress {
    unsigned int ipv4;
    unsigned char bytes[4];
};

int main() {
    union IPAddress ip;
    ip.bytes[0] = 192;
    ip.bytes[1] = 168;
    ip.bytes[2] = 1;
    ip.bytes[3] = 100;

    printf("IPv4 Address: %d.%d.%d.%d\n", ip.bytes[0], ip.bytes[1],
ip.bytes[2], ip.bytes[3]);

    return 0;
}
```

19. Union for Student Information

```
#include <stdio.h>
union StudentInfo {
```

```
    char name[30];
    struct {
        int age;
        float gpa;
    } details;
};

int main() {
    union StudentInfo student;
    strcpy(student.name, "Alice");
    printf("Name: %s\n", student.name);

    student.details.age = 20;
    student.details.gpa = 3.75;
    printf("Age: %d, GPA: %.2f\n", student.details.age,
student.details.gpa);

    return 0;
}
```

20. Union for Language Selection

```
#include <stdio.h>

union LanguageChoice {
    int option;
    char lang[20];
};

int main() {
    union LanguageChoice lc;
    lc.option = 1;

    if (lc.option == 1) {
        strcpy(lc.lang, "English");
    } else if (lc.option == 2) {
        strcpy(lc.lang, "Spanish");
    } else {
```

```
        strcpy(lc.lang, "Unknown");
    }

    printf("Selected language: %s\n", lc.lang);

    return 0;
}
```

21. Union for Temperature Conversion (Kelvin, Celsius, Fahrenheit)

```
#include <stdio.h>

union Temperature {
    float celsius;
    float fahrenheit;
    float kelvin;
};

int main() {
    union Temperature temp;
    temp.celsius = 25.0;
    printf("Temperature in Celsius: %.2f°C\n", temp.celsius);

    temp.fahrenheit = (temp.celsius * 9 / 5) + 32;
    printf("Temperature in Fahrenheit: %.2f°F\n", temp.fahrenheit);

    temp.kelvin = temp.celsius + 273.15;
    printf("Temperature in Kelvin: %.2fK\n", temp.kelvin);

    return 0;
}
```

22. Union for Vehicle Information (Car, Bike)

```c
#include <stdio.h>

union Vehicle {
    struct {
        char make[20];
        int year;
    } car;
    struct {
        char brand[20];
        int year;
    } bike;
};

int main() {
    union Vehicle v;
    strcpy(v.car.make, "Toyota");
    v.car.year = 2023;
    printf("Car: %s, Year: %d\n", v.car.make, v.car.year);

    strcpy(v.bike.brand, "Trek");
    v.bike.year = 2022;
    printf("Bike: %s, Year: %d\n", v.bike.brand, v.bike.year);

    return 0;
}
```

23. Union for Music Track Information (Song, Podcast)

```c
#include <stdio.h>
#include<string.h>
union TrackInfo {
    struct {
        char title[50];
        char artist[30];
        int duration;
    } song;
    struct {
```

```
        char title[50];
        char host[30];
        int duration;
    } podcast;
};

int main() {
    union TrackInfo track;
    strcpy(track.song.title, "Imagine");
    strcpy(track.song.artist, "John Lennon");
    track.song.duration = 180;
    printf("Song: %s by %s, Duration: %d seconds\n", track.song.title,
track.song.artist, track.song.duration);

    strcpy(track.podcast.title, "Tech Talk");
    strcpy(track.podcast.host, "Alice Smith");
    track.podcast.duration = 1200;
    printf("Podcast: %s hosted by %s, Duration: %d seconds\n",
track.podcast.title, track.podcast.host, track.podcast.duration);

    return 0;
}
```

24. Union for Geographical Coordinates (Latitude, Longitude)

```
#include <stdio.h>

union Coordinates {
    struct {
        float latitude;
        float longitude;
    } point;
};

int main() {
    union Coordinates loc;
    loc.point.latitude = 34.0522;
    loc.point.longitude = -118.2437;
```

```
    printf("Latitude: %.4f, Longitude: %.4f\n", loc.point.latitude,
loc.point.longitude);

    return 0;
}
```

25. Union for Employee Information (Full-time, Part-time)

```
#include <stdio.h>
#include<string.h>
union EmployeeInfo {
    struct {
        char name[50];
        int employeeID;
        float salary;
    } fullTime;
    struct {
        char name[50];
        int employeeID;
        int hoursWorked;
    } partTime;
};

int main() {
    union EmployeeInfo emp;
    strcpy(emp.fullTime.name, "John Doe");
    emp.fullTime.employeeID = 101;
    emp.fullTime.salary = 50000.0;
    printf("Full-time Employee: %s, ID: %d, Salary: $%.2f\n",
emp.fullTime.name, emp.fullTime.employeeID, emp.fullTime.salary);

    strcpy(emp.partTime.name, "Alice Smith");
    emp.partTime.employeeID = 102;
    emp.partTime.hoursWorked = 20;
    printf("Part-time Employee: %s, ID: %d, Hours Worked: %d\n",
emp.partTime.name, emp.partTime.employeeID, emp.partTime.hoursWorked);
```

```
    return 0;
}
```

26. Union for User Authentication (Username or Email)

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

union AuthInfo {
    char username[50];
    char email[50];
};

int main() {
    union AuthInfo user;
    strcpy(user.username, "john_doe");
    printf("Username: %s\n", user.username);

    strcpy(user.email, "john@example.com");
    printf("Email: %s\n", user.email);

    return 0;
}
```

27. Union for Shape Calculation (Circle, Rectangle)

```
#include <stdio.h>

union Shape {
    struct {
        float radius;
    } circle;
    struct {
        float length;
        float width;
    } rectangle;
```

```
};

int main() {
    union Shape s;
    s.circle.radius = 5.0;
    printf("Circle Area: %.2f\n", 3.14 * s.circle.radius *
s.circle.radius);

    s.rectangle.length = 4.0;
    s.rectangle.width = 6.0;
    printf("Rectangle Area: %.2f\n", s.rectangle.length *
s.rectangle.width);

    return 0;
}
```

28. Union for Book Information (Fiction, Non-fiction)

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

union Book {
    struct {
        char title[50];
        char author[30];
    } fiction;
    struct {
        char title[50];
        char subject[30];
    } nonfiction;
};

int main() {
    union Book b;
    strcpy(b.fiction.title, "The Great Gatsby");
    strcpy(b.fiction.author, "F. Scott Fitzgerald");
    printf("Fiction Book: %s by %s\n", b.fiction.title, b.fiction.author);
```

```
    strcpy(b.nonfiction.title, "The Selfish Gene");
    strcpy(b.nonfiction.subject, "Biology");
    printf("Non-fiction Book: %s (Subject: %s)\n", b.nonfiction.title,
b.nonfiction.subject);

    return 0;
}
```

29. Union for Color Representation (RGB and Hex)

```
#include <stdio.h>

union Color {
    struct {
        unsigned char red;
        unsigned char green;
        unsigned char blue;
    } rgb;
    unsigned int hex;
};

int main() {
    union Color c;
    c.rgb.red = 255;
    c.rgb.green = 0;
    c.rgb.blue = 0;
    printf("RGB: (%d, %d, %d)\n", c.rgb.red, c.rgb.green, c.rgb.blue);

    c.hex = 0x00FF00; // Green in hex
    printf("Hex: 0x%06X\n", c.hex);

    return 0;
}
```

30. Union for Currency Conversion (USD, EUR, GBP)

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

```c
union Currency {
    double usd;
    double eur;
    double gbp;
};

int main() {
    union Currency money;
    money.usd = 100.0;
    printf("USD: $%.2f\n", money.usd);

    money.eur = money.usd * 0.85;
    printf("EUR: €%.2f\n", money.eur);

    money.gbp = money.usd * 0.75;
    printf("GBP: £%.2f\n", money.gbp);

    return 0;
}
```

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Happy learning ^^