Опис Union structure (C language).

Union - це спеціальний тип даних, доступний на мові C, який дозволяє зберігати різні типи даних в одному і тому ж місці пам'яті. Ви можете визначити об'єднання з багатьма членами, але лише один член може містити значення в будь-який момент часу. Профспілки забезпечують ефективний спосіб використання одного і того ж місця в пам'яті для багатоцільового використання.

Щоб визначити Union, ви повинні використовувати оператор об’єднання так само, як і під час визначення структури. У заяві про об'єднання визначається новий тип даних із кількома членами для вашої програми. Формат заяви про спілку такий:

union [union tag] {  
member definition;  
member definition;  
...  
member definition;  
} [one or more union variables];

union tag є необов’язковим, і кожне визначення члена є звичайним визначенням змінної, таким як int i; або float f; або будь-яке інше дійсне визначення змінної. В кінці визначення об'єднання, перед заключною крапкою з комою, ви можете вказати одну або кілька змінних об'єднання, але це необов'язково. Ось спосіб, яким ви могли б визначити тип об’єднання з іменем Дані, що має три члени i, f та str -

union Data {  
 int i;  
 float f;  
 char str[20];  
} data;

Тепер змінна типу Data може зберігати ціле число, число з плаваючою точкою або рядок символів. Це означає, що одна змінна, тобто одне і те ж місце пам'яті, може використовуватися для зберігання декількох типів даних. Ви можете використовувати будь-які вбудовані або визначені користувачем типи даних всередині об’єднання відповідно до ваших вимог.

Пам’ять, яку займає об’єднання, буде достатньо великою, щоб вмістити найбільшого члена об’єднання. Наприклад, у наведеному вище прикладі тип даних займе 20 байт пам'яті, оскільки це максимальний простір, який може займати рядок символів. У наведеному нижче прикладі відображається загальний обсяг пам’яті, зайнятий вищевказаним об’єднанням -

#include <stdio.h>  
#include <string.h>  
  
union Data {  
 int i;  
 float f;  
 char str[20];  
};  
  
int main( ) {  
 union Data data;  
 printf( "Memory size occupied by data : %d\n", sizeof(data));  
 return 0;  
}

Коли вищезгаданий код компілюється та виконується, він дає такий результат -

Memory size occupied by data : 20

Для доступу до будь-якого члена union ми використовуємо оператора доступу членів (.). Оператор доступу члена кодується як період між ім’ям змінної об’єднання та членом об’єднання, до якого ми хочемо отримати доступ. Використовувається ключове слово union для визначення змінних типу union. Наступний приклад показує, як використовувати спілки в програмі

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

Коли вищезгаданий код компілюється та виконується, він дає такий результат -

data.i : 1917853763

data.f : 4122360580327794860452759994368.000000

data.str : C Programming

Тут ми бачимо, що значення i та f членів об'єднання були пошкоджені, оскільки остаточне значення, присвоєне змінній, займало місце пам'яті, і це є причиною того, що значення елемента str надрукується дуже добре.

А тепер давайте ще раз розглянемо той самий приклад, де ми будемо використовувати по одній змінній за раз, що є основною метою об’єднання -

#include <stdio.h>  
#include <string.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 : %f\n", data.f);  
  
 strcpy( data.str, "C Programming");  
 printf( "data.str : %s\n", data.str);  
  
 return 0;  
}

Коли вищезгаданий код компілюється та виконується, він дає такий результат -

data.i : 10

data.f : 220.500000

data.str : C Programming

Також Union може містити декілька структур даних у собі, і у потрібний момент можна до них звертатися, присвоювати/діставати значення змінних структур, побачимо на прикладі -

#include<stdio.h>  
int main() {  
 struct student {  
 char name[30];  
 int rollno;  
 float percentage;  
 };  
 union details {  
 struct student s1;  
 };  
 union details set;  
 scanf("%s", &set.s1.name);

scanf("%d", &set.s1.rollno);

scanf("%f", &set.s1.percentage);

printf("\name : %s", set.s1.name);

printf("\nRollno : %d", set.s1.rollno);

printf("\nPercentage : %f", set.s1.percentage);

Коли вищезгаданий код компілюється та юзер вводить дані, у терміналі він побачить введені дані.