Techimax

Fundamentals of Programming in C++

Unions

```
union union_name {
    member definition;
}
```

Union is a user-defined datatype. All the members of union **share same memory location**. Size of union is decided by the **size of largest member** of union. If you want to use same memory location for two or more members, union is the best for that.

```
union test {
    int x, y;
};

int main()
{
    // A union variable t
    union test t;

    // t.y also gets value 2
    t.x = 2;

    cout << "After making x = 2:\n"
         << "x = " << t.x
         << ", y = " << t.y
         << endl;

    cout << "Size of test : " << sizeof(t);
}
```

```
After making x = 2:
x = 2, y = 2
Size of test : 4
```

Enums

```
enum enumerated_type_name{value1, value2, value3…..valueN};
```

```
enum week
{
    Mon, Tue, Wed, Thur,
    Fri, Sat, Sun
};

int main()
{
    enum week day;
    day = Wed;
    cout << day;
    // 2
}
```

Enumeration (or enum) is a user defined data type. It is mainly used to assign **names to integral constants**, the names make a program easy to read and maintain.

```
enum State {Working = 1, Failed = 0, Freezed = 0};

int main()
{
   printf("%d, %d, %d", Working, Failed, Freezed);
  // 1, 0, 0
   return 0;
}
```

Typedef

```
typedef type name;
```

typedef keyword is used to assign a new name to any **existing** data-type.

```
typedef unsigned char BYTE;

int main()
{
    BYTE b1, b2;
    b1 = 'c';
    cout << b1;
    //c
    return 0;
}
```

#define

```
#define identifier replacement
```

Macros are a piece of code in a program which is given some name. Whenever this name is encountered by the compiler the compiler replaces the name with the actual piece of code.

The '#define' directive is used to define a macro.

```
#define LIMIT 5
int main()
{
    for (int i = 0; i < LIMIT; i++)
    {
        cout << i << "\n";
    }
}
```

```
#define getMax(a, b) (a > b ? a : b)
int main()
{
    cout << getMax(5, 7);
    // 7
}
```

typedef vs #define

typedef

- typedef is limited to giving symbolic names to **types only**
- interpretation is performed by the **compiler**
- should be **terminated with semicolon**
- typedef is the **actual definition** of a new type
- follows the **scope rule** which means if a new type is defined in a scope (inside a function), then the new type name will only be visible till the scope is there

#define

- #define can be used to define an **alias** for values as well, e.g., you can define 1 as ONE, 3.14 as PI, etc.
- #define statements are performed by **preprocessor**.
- should **not be terminated with a semicolon**
- #define will just **copy-paste** the definition values at the point of use
- when preprocessor encounters #define, it **replaces all the occurrences**, after that (No scope rule is followed).

Recursion

A function that calls itself is known as recursive function. And, this technique is known as recursion.

```
int factorial(int n)
{
    if (n ⩾ 1)
        return n * factorial(n - 1);
    else
        // exit condition
        return 1;
}

int main()
{
    cout << "fact of 6 :" << factorial(6);
    // 720
}
```