1. **Từ khóa extern, voltage và static**

Ref: <http://ktmt.github.io/blog/2013/04/06/tu-khoa-static-trong-c/>

* 1. **Static**
* Biến static
* Hàm static

Đặc điểm:

* Khi được sử dụng trong phạm vi toàn cục của 1 đơn vị biên dịch, static hạn chế truy cập từ các đơn vị biên dịch khác (áp dụng với cả hàm và biến toàn cục).
* Khi được sự dụng trong phạm vi cục bộ của 1 thủ tục hay hàm, static có nghĩa là biến được khai báo tồn tại trong suốt thời gian chạy của chương trình và chỉ được khởi tạo 1 lần duy nhất.

|  |
| --- |
| // nếu khai báo static trong hai hàm khác nhau thì kq đc cập nhật ở cái gần nhất ?  #include "main.h"  int sizeofx(int x)  {  **static int kq = 0;**  kq += sizeof(x);  printf("kq trong sizeofx: %d", kq);  return kq;  }  int main()  {  int counter = 0;  int a = sizeof(counter);  int b = sizeofx(counter);  int c = sizeofx(counter);  **static int kq = 1;**  printf("kq trong main: %d", kq);  return 0;  } |

* 1. **Volatile**
* Tác dụng: thông báo với trình biên dịch không tối ưu code với biến đó

Ví dụ:

|  |
| --- |
| while (a)// a is declared with volatile modifier { ..//there isn't code changing a } Với đoạn code trên mà a không được khai báo là volatile thì khi tối ưu hóa code, bộ dịch sẽ bỏ qua việc kiểm tra điều kiện vì trong vòng lặp không có đoạn code nào ảnh hưởng đến biến a. Nhưng nhiều khi biến a lại được can thiệp từ một thread khác, thế nên người ta cho thêm cái modifier volatile vào để thông báo cho bộ dịch được biết.  Link: <http://www.ddth.com/showthread.php/241484-extern-volatile> |

* 1. **Extern**

<http://www.cquestions.com/2011/02/extern-keyword-in-c.html>

* Một biến global mặc định là một biến extern và có giá trị 0 với kiểu int và NULL với các kiểu khác.
* Nhưng nếu khai báo extern cho biến đó thì lỗi do chưa đc khởi tạo giá trị. Muốn dùng thì phải khai báo và khởi tạo cùng lúc. Ví dụ:

|  |  |
| --- | --- |
| * (a) * #include <stdio.h> * int i;    //By default it is extern variable * int main(){ * printf("%d",i); * return 0; * } * Output: 0 | (b)  #include <stdio.h>  extern int i;    //extern variable  int main(){      printf("%d",i);      return 0;  }  Output: Compilation error, undefined symbol i. |

Question: In Both program variable i is extern variable. But why output is different? Read second and third points.

|  |
| --- |
| (c)  #include <stdio.h>  void sum(int,int) //By default it is extern.  int main(){      int a=5,b=10;      sum(a,b);      return 0;  }  void sum(int a,int b){      printf("%d”",a+b);  }  Output: 15 |

2. When we use extern modifier with any variables it is only declaration i.e. memory is not allocated for these variable. Hence in second case compiler is showing error unknown symbol i. To define a variable i.e. allocate the memory for extern variables it is necessary to initialize the variables. For example:

#include <stdio.h>

extern int i=10;    //extern variable **Khai báo + khởi tạo**

int main(){

    printf("%d",i);

    return 0;

}

Output: 10

3. If you will not use extern keyword with global variables then compiler will automatically initialize with default value to extern variable.

4. **Default initial value of extern integral type variable is zero otherwise null**. For example:

#include <stdio.h>

char c;

int i;

float f;

char \*str;

int main(){

    printf("%d %d %f %s",c,i,f,str);

    return 0;

}

Output: 0 0 0.000000 (null)

5. **We cannot initialize extern variable locally, only declare but must initialize var globally in some where** i.e. within any block either at the time of declaration or separately. **We can only initialize extern variable globally**. For example:

(a)

#include <stdio.h>

int main(){

extern int i=10; //Try to initialize extern variable

                 //locally.

    printf("%d",i);

    return 0;

}

Output: Compilation error: Cannot initialize extern variable.

(b)

#include <stdio.h>

int main(){

    extern int i; //Declaration of extern variable i.

    int i=10;     //Try to locally initialization of

                  //extern variable i.

    printf("%d",i);

    return 0;

}

Output: Compilation error: Multiple declaration of variable i.

6. **If we declare any variable as extern variable then it searches that variable either it has been initialized or not**. If it has been initialized which may be either extern or static\* then it is ok otherwise compiler will show an error. For example:

(a)

#include <stdio.h>

int main(){

    extern int i; //It will search the initialization of

                  //variable i.

    printf("%d",i);

    return 0;

}

int i=20;    //Initialization of variable i.

Output: 20

(b)

#include <stdio.h>

int main(){

extern int i; //It will search the any initialized

              //variable i which may be static or

              //extern.

printf("%d",i);

    return 0;

}

extern int i=20; //Initialization of extern variable i.

Output: 20

(c)

#include <stdio.h>

int main(){

extern int i; //It will search the any initialized

              //variable i which may be static or

              //extern.

    printf("%d",i);

    return 0;

}

static int i=20; //Initialization of static variable i.

Output: 20

(d)

#include <stdio.h>

int main(){

    extern int i;   //variable i has declared but not

                    //initialized

    printf("%d",i);

    return 0;

}

Output: Compilation error: Unknown symbol i.

7. A particular **extern variable can be declared many times** but **we can initialize at only one time**. For example:

(a)

extern int i; //Declaring the variable i.

int i=25;     //Initializing the variable.

extern int i; //Again declaring the variable i.

#include <stdio.h>

int main(){

    extern int i; //Again declaring the variable i.

    printf("%d",i);

    return 0;

}

Output: 25

(b)

extern int i; //Declaring the variable

int i=25;     //Initializing the variable

#include <stdio.h>

int main(){

         printf("%d",i);

    return 0;

}

int i=20; //Initializing the variable

Output: Compilation error: Multiple initialization variable i.

8. **We cannot write any assignment statement globally**. For example:

#include <stdio.h>

extern int i;

int i=10;   //Initialization statement

**i=25;       //Assignment statement**

int main(){

    printf("%d",i);

    return 0;

}

Output: Compilation error

Note: Assigning any value to the variable at the time of declaration is known as initialization while assigning any value to variable not at the time of declaration is known assignment.

(b)

#include <stdio.h>

extern int i;

int main(){

    i=25;       //Assignment statement

    printf("%d",i);

    return 0;

}

int i=10;   //Initialization statement

Output: 25

9. If declared an extern variables or function globally then its visibility will whole the program which may contain one file or many files. For example consider a c program which has written in two files named as one.c and two.c:

(a)

//one.c

#include<conio.h>

int i=25; //By default extern variable

int j=5;  //By default extern variable

/\*\*

Above two line is initialization of variable i and j.

\*/

void main(){

    clrscr();

    sum();

    getch();

}

//two.c

#include<stdio.h>

extern int i; //Declaration of variable i.

extern int j; //Declaration of variable j.

/\*\*

Above two lines will search the initialization statement of variable i and j either in two.c (if initialized variable is static or extern) or one.c (if initialized variable is extern)

\*/

void sum(){

    int s;

    s=i+j;

    printf("%d",s);

}

Compile and execute above two file one.

Note: In the above example function sum which was declared and defined in two.c has also storage class extern. So we can call from other file (one.c).If it will static then we cannot call function sum since static storage class is only visible to the file where it has declared.

10. An extern variables or functions have external linkage. An external linkage variables or functions are visible to all files.

1. **Phân biệt hàm và toán tử**

* Ví dụ: toán tử sizeof

An operator is compiled to a sequence of instructions by the compiler. But if the code calls a function, it has to jump to a separate piece of code.