Registers are faster than memory to access, so the variables which are most frequently used in a C program can be put in registers using *register* keyword. The keyword *register* hints to compiler that a given variable can be put in a register. It's compiler's choice to put it in a register or not. Generally, compilers themselves do optimizations and put the variables in register.

If you use & operator with a register variable then compiler may give an error or warning (depending upon the compiler you are using), because when we say a variable is a register, it may be stored in a register instead of memory and accessing address of a register is invalid. Try below program.

```
int main()
{
  register int i = 10;
  int *a = &i;
  printf("%d", *a);
  getchar();
  return 0;
}
```

register keyword can be used with pointer variables. Obviously, a register can have address of a memory location. There would not be any problem with the below program.

```
int main()
{
  int i = 10;
  register int *a = &i;
  printf("%d", *a);
  getchar();
  return 0;
}
```

Register is a storage class, and C doesn’t allow multiple storage class specifiers for a variable. So, *register* can not be used with *static* . Try below program.

```
int main()
{
  int i = 10;
  register static int *a = &i;
  printf("%d", *a);
  getchar();
  return 0;
}
```

There is no limit on number of register variables in a C program, but the point is compiler may put some variables in register and some not.

Thank you for watching!

Please leave us your comments.