```
#include <stdio.h>

void add(const double [], double []);
void add_res(double * restrict a, double * restrict b);

int main()
{
  double a[] = {1, 2, 3, 4};

  add(a, &a[1]);

  for(int k = 0; k < 4; k++)
    printf("%8.0f", a[k]);
  printf("\n");

  add_res(a, &a[1]);

  for(int k = 0; k < 4; k++)
    printf("%8.0f", a[k]);
  printf("\n");

  return 0;
}

void add(const double a[], double b[])
{
  b[0] = 1111;
}

void add_res(double * restrict a, double * restrict b)
{
  b[0] = 2222;
}
```

```
% icc -std=c99 add2.c
% a.out
       1    1111       3       4
       1    2222       3       4

Change
void add(const double a[], double b[])
{
  b[0] = 1111;
}

to

void add(const double a[], double b[])
{
  a[0] = 1111;
}

% icc -std=c99 add2.c
add2.c(27): error #137: expression must be a modifiable lvalue
    a[0] = 1111;
    ^

compilation aborted for add2.c (code 2)
```

```
void horner(double px[], const double x[], const double coeff[], int n)
{
  double          xj;

  for (int j = 0; j < n; j++) {
    xj    = x[j];
    px[j] = coeff[0] + xj*(coeff[1] + xj*(coeff[2] + xj*(coeff[3] + xj*coeff[4])));
  }
}

% icc -S -O3 horner.c
PART of horner.s

.L2:                                                                    #
        fstpl     -8(%ecx,%eax,8)                                       #
.L1:                                                                    #
        fldl      (%edi,%eax,8)                                         #6.13
        fldl      32(%esi)                                              #8.36
        fmul      %st(1), %st                                           #8.36
        faddl     24(%esi)                                              #8.36
        fmul      %st(1), %st                                           #8.36
        faddl     16(%esi)                                              #8.36
        fmul      %st(1), %st                                           #8.36
        faddl     8(%esi)                                               #8.36
        fmulp     %st, %st(1)                                           #8.36
        faddl     (%esi)                                                #8.36
        addl      $1, %eax                                              #5.26
        cmpl      %edx, %eax                                            #5.3
        jl        .L2           # Prob 97%                              #5.3
        fstpl     -8(%ecx,%eax,8)                                       #7.5
```

```
% icc -S -O3 -fno-alias horner.c
.L2:                                                                    #
        fstpl     -8(%esi,%eax,8)                                       #
.L1:                                                                    #
        fld       %st(0)                                                #8.36
        fldl      (%ecx,%eax,8)                                         #6.13
        fmul      %st, %st(1)                                           #8.36
        fxch      %st(1)                                                #8.36
        fadd      %st(3), %st                                           #8.36
        fmul      %st(1), %st                                           #8.36
        fadd      %st(4), %st                                           #8.36
        fmul      %st(1), %st                                           #8.36
        fadd      %st(5), %st                                           #8.36
        fmulp     %st, %st(1)                                           #8.36
        fadd      %st(5), %st                                           #8.36
        addl      $1, %eax                                              #5.26
        cmpl      %edx, %eax                                            #5.3
        jl        .L2           # Prob 97%                              #5.3
        fstpl     -8(%esi,%eax,8)                                       #7.5

Fortran gives the same type of code as does the use of restricted pointers.
gcc does not gives faster code for the restricted case.
```