

```

entry:
  %retval = alloca i32, align 4
  %A = alloca [10 x i32], align 16
  %B = alloca [10 x i32], align 16
  %i = alloca i32, align 4
  %j = alloca i32, align 4
  store i32 0, i32* %retval, align 4
  %0 = bitcast [10 x i32]* %A to i8*
  call void @llvm.memcpy.p0i8.p0i8.i64(i8* align 16 %0, i8* align 16 bitcast
... ([10 x i32]* @ _const.main.A to i8*), i64 40, i1 false)
  %1 = bitcast [10 x i32]* %B to i8*
  call void @llvm.memset.p0i8.i64(i8* align 16 %1, i8 0, i64 40, i1 false)
  store i32 0, i32* %j, align 4
  store i32 0, i32* %i, align 4
  br label %for.cond

```

```

for.cond:
  %2 = load i32, i32* %i, align 4
  %cmp = icmp slt i32 %2, 10
  br i1 %cmp, label %for.body, label %for.end, !prof !34

```

T

F

```

for.body:
  %3 = load i32, i32* %j, align 4
  %idxprom = sext i32 %3 to i64
  %arrayidx = getelementptr inbounds [10 x i32], [10 x i32]* %A, i64 0, i64
... %idxprom
  %4 = load i32, i32* %arrayidx, align 4
  %mul = mul nsw i32 %4, 11
  %5 = load i32, i32* %i, align 4
  %add = add nsw i32 %mul, %5
  %6 = load i32, i32* %i, align 4
  %idxprom1 = sext i32 %6 to i64
  %arrayidx2 = getelementptr inbounds [10 x i32], [10 x i32]* %B, i64 0, i64
... %idxprom1
  store i32 %add, i32* %arrayidx2, align 4
  %7 = load i32, i32* %i, align 4
  %cmp3 = icmp slt i32 %7, 8
  br i1 %cmp3, label %if.then, label %if.end, !prof !35

```

T

F

```

if.then:
  %8 = load i32, i32* %i, align 4
  store i32 %8, i32* %j, align 4
  br label %if.end

```

```

if.end:
  %9 = load i32, i32* %i, align 4
  %idxprom4 = sext i32 %9 to i64
  %arrayidx5 = getelementptr inbounds [10 x i32], [10 x i32]* %B, i64 0, i64
... %idxprom4
  %10 = load i32, i32* %arrayidx5, align 4
  %call = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([4 x
... i8], [4 x i8]* @.str, i64 0, i64 0), i32 noundef %10)
  br label %for.inc

```

```

for.inc:
  %11 = load i32, i32* %i, align 4
  %inc = add nsw i32 %11, 1
  store i32 %inc, i32* %i, align 4
  br label %for.cond, !llvm.loop !36

```

```

for.end:
  ret i32 0

```

CFG for 'main' function