```
#!/usr/local/bin/perl -w

=head1 NAME

ack - grep-like text finder for large trees of text

=head1 DESCRIPTION

F<ack> is a F<grep>-like program with optimizations for searching through
large trees of source code.

Key improvements include:

=over 4

=item * Defaults to only searching program source code

=item * Defaults to recursively searching directories

=item * Uses Perl regular expressions

=item * Highlights matched text

=back

=cut

use strict;
use App::Ack;
use File::Find;
use Term::ANSIColor;
use Getopt::Long;

our $is_tty = -t STDOUT;

our $opt_group =    $is_tty;
our $opt_color =    $is_tty;
our $opt_all =      0;
our $opt_help =     0;
our %lang;

our @languages_supported = qw( cc perl php python ruby shell sql );

my %options = (
    h           => \( our $opt_h = 0 ),
    i           => \( our $opt_i = 0 ),
    l           => \( our $opt_l = 0 ),
    "m=i"       => \( our $opt_m = 0 ),
    n           => \( our $opt_n = 0 ),
    o           => \( our $opt_o = 0 ),
    v           => \( our $opt_v = 0 ),
    w           => \( our $opt_w = 0 ),
    "all!"      => \$opt_all,
    "group!"    => \$opt_group,
    "color!"    => \$opt_color,
    "help"      => \$opt_help,
    "version"   => sub { print "ack $App::Ack::VERSION\n" and exit 1; },
);
for my $i ( @languages_supported ) {
    $options{ "$i!" } = \$lang{ $i };
}

GetOptions( %options ) or $opt_help = 1;

my $languages_supported_set =   grep { defined $lang{$_} && ($lang{$_} == 1) } @lan
guages_supported;
```

```
my $languages_supported_unset = grep { defined $lang{$_} && ($lang{$_} == 0) } @lan
guages_supported;

# If anyone says --noperl, we assume all other languages must be on.
if ( !$languages_supported_set ) {
    for ( keys %lang ) {
        $lang{$_} = 1 unless defined $lang{$_};
    }
}

if ( $opt_help || !@ARGV ) {
    print <DATA>;  # Show usage
    exit 1;
}

my $re = shift or die "No regex specified\n";

if ( $opt_w ) {
    $re = $opt_i ? qr/\b$re\b/i : qr/\b$re\b/;
}
else {
    $re = $opt_i ? qr/$re/i : qr/$re/;
}

my @what = @ARGV ? @ARGV : ".";
find( \&handler, @what );

sub handler {
    return if /~$/;

    if ( -d ) {
        $File::Find::prune = 1 if $_ eq 'CVS';
        $File::Find::prune = 1 if $_ eq '.svn';
        $File::Find::prune = 1 if $_ eq 'blib';
        $File::Find::prune = 1 if $_ eq 'RCS';
        $File::Find::prune = 1 if $opt_n && ( $_ ne "." );
        return;
    }

    if ( !$opt_all ) {
        my $type = filetype( $_ );

        return unless defined $type;
        return unless $lang{$type};
    }

    search( $_, $File::Find::name, $re );
}


sub search {
    my $filename = shift;
    my $dispname = shift;
    my $regex = shift;

    my $nmatches = 0;

    local $_;
    open( my $fh, $filename ) or die "Can't open $filename: $!";
    while (<$fh>) {
        if ( /$re/ ) {
            ++$nmatches;

            if ( $opt_l ) {
                print "$dispname\n";
                last;
```

```
            }

            # No point in still searching if we know we want negative match
            last if $opt_v;

            my $out;

            if ( $opt_o ) {
                $out = "$&\n";
            }
            else {
                $out = $_;
                $out =~ s/($re)/colored($1,"black on_yellow")/eg if $opt_color;
            }

            if ( $opt_h ) {
                print $out;
            }
            else {
                my $colorname = $opt_color ? colored( $dispname, "bold green" ) : $
dispname;
                if ( $opt_group ) {
                    print "$colorname\n" if $nmatches == 1;
                    print "$.:$out";
                }
                else {
                    print "$colorname:$.:$out";
                }
            }

            last if $opt_m && ( $nmatches >= $opt_m );
        } # match
    } # while

    print "$dispname\n" if $opt_v && !$nmatches;
    print "\n" if $nmatches && ($opt_group && !$opt_l);

    close $fh;
}


=head1 TODO

=over 4

=item * Search through standard input if no files specified

=item * Add a --[no]comment option to grep inside or exclude comments.

=back

=cut

1;

__DATA__
Usage: ack [OPTION]... PATTERN [FILES]
Search for PATTERN in each source file in the tree from cwd on down.
If [FILES] is specified, then only those files/directories are checked.

Example: ack -i select

Output & searching:
    -i              ignore case distinctions
    -v              invert match: select non-matching lines
    -w              force PATTERN to match only whole words
```

```
    -l                 only print filenames containing matches
    -o                 show only the part of a line matching PATTERN
    -m=NUM             stop after NUM matches
    -h                 don't print filenames
    --[no]group        print a blank line between each file's matches
                       (default: on unless output is redirected)
    --[no]color        highlight the matching text (default: on unless
                       output is redirected)

File selection:
    -n                 No descending into subdirectories
    --[no]cc           .c and .h                          (default: on)
    --[no]perl         .pl, .pm, .pod, .t, .tt and .ttml  (default: on)
    --[no]php          .html, .php, and .phpt             (default: on)
    --[no]python       .py                                (default: on)
    --[no]ruby         .rb                                (default: on)
    --[no]shell        shell scripts                      (default: on)
    --[no]sql          .sql and .ctl files                (default: on)
    --all              All files, regardless of extension
                       (but still skips CVS, .svn and blib dirs)
```