type is not itself a pointer, but you only can assign the address of non-\texttt{const} data to a non-\texttt{const} pointer.

Suppose you have an array of \texttt{const} data:

\begin{verbatim}
\end{verbatim}

The prohibition against assigning the address of a constant array means that you cannot pass the array name as an argument to a function using a non-constant formal argument:

\begin{verbatim}
int sum(int arr[], int n);  // should have been const int arr[]
...
int j = sum(months, 12);    // not allowed
\end{verbatim}

The function call attempts to assign a \texttt{const} pointer (\texttt{months}) to a non-\texttt{const} pointer (\texttt{arr}), and the compiler disallows the function call.

\textbf{Use \texttt{const} When You Can}

There are two strong reasons to declare pointer arguments as pointers to constant data:

- This protects you against programming errors that inadvertently alter data.

- Using \texttt{const} allows a function to process both \texttt{const} and non-\texttt{const} actual arguments, whereas a function omitting \texttt{const} in the prototype only can accept non-\texttt{const} data.

You should declare formal pointer arguments as pointers to \texttt{const} whenever it's appropriate to do so.

Now another subtle point: the declarations

\begin{verbatim}
int age = 39;
const int * pt = &age;
\end{verbatim}