My somewhat elliptic title refers, of course, to the programmer; so much you may have guessed. What, in all probability, you could not have guessed is that I have chosen to use the words “craftsman” and “scientist” in a very specific meaning: they have been chosen to characterize the results of two extreme techniques of education, and this luncheon speech will be devoted to a (be it short) discussion of their role in the education of programmers, in the teaching of programming. For the transmission of knowledge and skills both techniques have been used side by side since many centuries.

The future craftsman joins a master for seven meagre years, he works as an apprentice under his guidance and supervision, absorbing gradually, by osmosis so to speak, the skills of the craft, until he may be called a master himself. Craftsmen typically form Guilds and the guild members tend to keep their common craft as a well-guarded secret among themselves: not blowing the gaff is one of their rules of professional conduct. Note, finally, that old crafts have been lost, dependent as their survival was on the continuing transmission from one generation to the next.

The future scientist learns his trade as a student from a teacher, who, in contrast to the master who transfers his knowledge implicitly to his apprentice, tries to formulate the knowledge and to describe the skills as explicitly as possible, thereby bringing both into the public domain. The latter technique is the prevailing one at the Universities. It is no coincidence that the rise of the Universities occurred when the printing press became widely established, and it is no accident that each University regarded its
Library as its greatest treasure: the library was the embodiment of its specific calling. Scientists regard the free interchange of knowledge and insights as essential, and, in consequence, being non-secretive is one of their rules of professional conduct.

To this very day, both techniques are applied side by side: physicists, for instance, are mostly scientific, physicians, however, are mostly much more like guild members. Mathematicians are somewhere in between: mathematical results are published and taught quite openly, but there is very little explicit teaching on how to do mathematics, and publishing besides the results also the heuristics that led to them is regarded by many as “unscientific” and, therefore, bad style. Quite often the editor’s censorship will try to prohibit their publication.

I have sketched for you two extreme educational techniques, but this was only preparation: my real topic is “Where along this scale should we place the teaching of programming?”. This, as I have learned by sad experience, is a risky subject to discuss, because one always discusses it with people who themselves are involved in one way or another in the programming profession, and their personal involvement tends to evoke strong emotional reactions. Let us try to understand them, for only then we may be able to cope with them.

To make implicit knowledge explicit and to discuss how to describe skills, so that they can be transferred, implies, if not the birth at least the conception of a new science. But we should realize that changing a craft into a science, and making public property of the secret knowledge of the guild will always cause the guild members to feel threatened. For many a “puzzle-minded” virtuoso coder of the early sixties, the scientific development of the last decade has been most unwelcome. He feels like the medieval painter that could create a masterpiece whenever his experience enabled him to render proportions well, who suddenly found himself overtaken by all sorts of youngsters, pupils of Albrecht Dürer and the like, who had been taught the mathematical constructions that were guaranteed to surpass his most successful, but intuitive, renderings. And with nostalgia he looks back to the good old days when his experience and feeling made him an outstanding craftsman. And we should realize that, as far as programming is concerned, the battle is still going on. From a European country, the name of which I shall not divulge in order to avoid personal complications, I recently studied a proposal for the organization of its computing science teaching at University level. The majority of its authors—all of them professors of computing science in their country—should be characterized as “craftsmen”. As a result, their proposal had a pronounced anti-intellectualistic flavour: it stressed that the students should be taught how to solve the problems of “the real world” and that, therefore, the curriculum should pay as little attention as possible to “abstract subjects”. Such utterances are unmistakable and, undoubtedly, you recognize them. So much for the pure craftsman’s point of view.