This workshop on an important area of CAD/CAM and robotics focused primarily on industrial applications. Papers and sessions covered such topics as panoramic views of vision, systems for prototyping inspection in lithographic technologies, inspection technologies and applications, 3D vision techniques, and robot vision systems. 236 pp.

Order #412


Raleigh, NC, May 1982

Members—$18.75
Nonmembers—$25.00

Use order form on p. 97.

First presented at Compsac 81, this tutorial describes the major issues surrounding the improvement of software productivity. It also looks at the environmental aspects of software development and analyzes the impact of working conditions and tools on human performance in computer programming. The 39 articles included fall into such categories as programming measurements, requirements, and environments, plus alternatives to programming. 440 pp.

Order #391

TUTORIAL—PROGRAMMING PRODUCTIVITY: ISSUES FOR THE EIGHTIES

Edited by Capers Jones, 1981

Members—$21.00
Nonmembers—$28.00

Figure 9. A typical C program in an elementary typographic form. It uses fixed-width characters of a single size and a typeface with limited horizontal spacing variation.

Figure 10. A prototypical black-and-white visualization that would require a high-resolution, bit-map display terminal or very high-resolution hard-copy device to produce. The actual image was generated on a computer-controlled phototypesetter, a rare but not unheard of hard-copy device. The image illustrates the potential of a graphic design approach to textual program visualization.


calc1 /* reverse Polish desk calculator */
    
    int type;  
    char *getop(), *getop2();
    double op2, op2f, pop(), push();
    while ((type = getop()) != EOF) {  
        switch (type) {  
            case NUMBER:  
                push(op2f());  
                break;
            case '+':  
                push(pop1() + pop1());  
                break;
            case '-':  
                push(pop1() - pop1());  
                break;
            case '*':  
                push(pop1() * pop1());  
                break;
            case '/':  
                if (pop2f() == 0.0)  
                    push(pop1() / pop2f());  
                else  
                    printf( "divide by zero!");  
                    break;
            case 'I':  
                printf("%f, push(pop1());  
                break;
            case 'C':  
                clear();  
                break;
            case 'T':  
                printf("%s, push(pop1());  
                break;
            default:  
                printf("unknown command %s, type");  
                break;
        }
    }