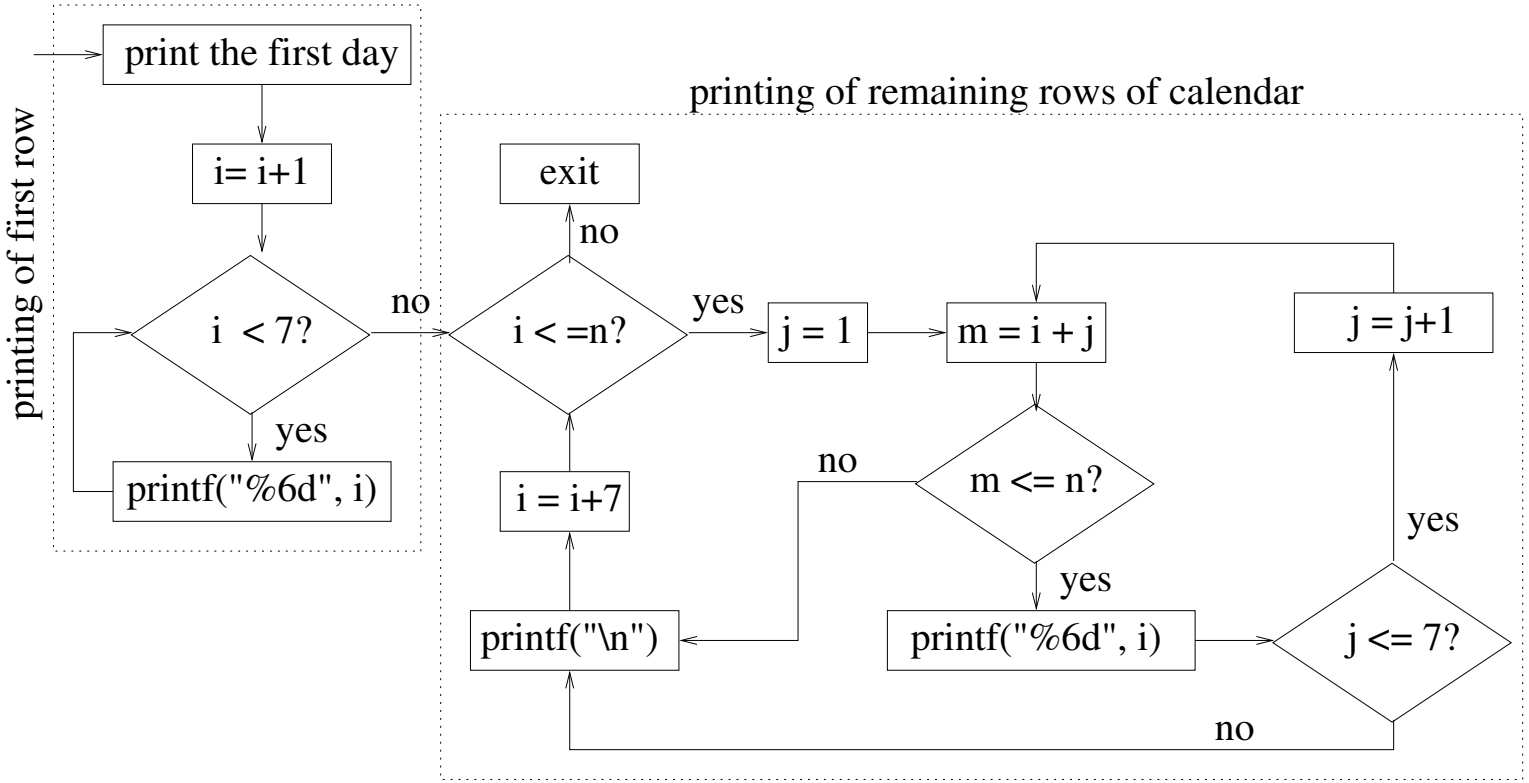
More Example Using Loops

Example 35 (contd)



More Example Using Loops

Example 35 (contd)

```
#include <stdio.h>
int main() {
    int i = 0;
    int n, m, day, j, w;

    printf("Enter number of days in the month: ");
    scanf("%d", &n);

    printf("Enter day: ");
    scanf("%d", &day);

    printf("   Mon   Tue   Wed   Thu   Fri   Sat   Sun\n\n");

    /* Print first row of the calendar, and subsequent rows
     * of the calendar*/
}
```

More Example Using Loops

Example 36 (contd)

```
// Print the first row of the calendar
switch (day) {
   case 1 : w = 6; printf("%*d", w, ++i);
            while (i < 7) printf("%6d", ++i); break;
   case 2 : w = 12; printf("%*d", w, ++i);
            while (i < 6) printf("%6d", ++i); break;
   case 3 : w = 18; printf("%*d", w, ++i);
            while (i < 5) printf("%6d", ++i); break;
   case 4 : w = 24; printf("%*d", w, ++i);
            while (i < 4) printf("%6d", ++i); break;
   case 5 : w = 30; printf("%*d", w, ++i);
            while (i < 3) printf("%6d", ++i); break;
   case 6 : w = 36; printf("%*d", w, ++i);
            while (i < 2) printf("%6d", ++i); break;
   case 7 : w = 42; printf("%*d", w, ++i); break;

   default : printf("Invalid day\n");  return -1;
}
```

More Example Using Loops

Example 36 (contd)

```
// Finish the printing of the first row.
printf("\n");

// Print rest of the other rows of the calendar
for (; i<=n; i=i+7) {
    for (j = 1; j <= 7; j++) {
        m =i+j;
        if (m <= n)
            printf("%6d",m);
        else break;
    }
    printf("\n");
}
```

Programming with Character Types

Scanf

- Format specification `%c` in `scanf` allows to read character.
- Similarly `%c` in `printf` allow character printing.
- `scanf` does n't skip white spaces.
- To skip white spaces, `scanf(" %c", &ch);` should be used.
- End of line can be detected by checking if character read is `'\n'`

```
do {
    scanf("%c", &ch);
} while (ch != '\n'); // detecting end of line
```

Programming with Character Types

Using getcar and putchar

- C provides functions for single character read/write.
- `getchar()` and `putchar()`.
- `ch = getchar();` reads one character.
- `getchar()` returns an `int`, so can be stored in an `int` variable.
- These functions are simpler and efficient (`scanf` and `printf` are generic).

```
do {
    ch = getchar();
} while (ch != '\n'); // detecting end of line using getchar()

while ((ch = getchar()) != '\n'); // simpler version

while ((ch = getchar()) == ' '); // skipping spaces
```

Programming with Character Types

Using getcar and putchar

- Mixing `getchar()` and `scanf` is not advisable.
- It could create problems as follows:

```
printf("Enter an integer: ");

/* Scanf would read i but peek next character (not consumed) */
scanf("%d", &i);

printf("Enter a command: ");

// Returns new line or leftover character
command = getchar();
```

Programming with Character Types

Example: Length of a Line

```
#include <stdio.h>
int main() {
    char ch;
    int length = 0;

    printf("Enter a message: ");

    while((ch = getchar()) != '\n') length++;
    printf("Length of the message: %d\n", length);
}
```

Programming with Character Types

Example (contd)

```
#include <stdio.h>
int main() {
    char ch, nV = 0, nO = 0;

    printf("Enter a message: ");

    while((ch = getchar()) != '\n') {
        switch (ch) {
            case 'a' :
            case 'e' :
            case 'i' :
            case 'o' :
            case 'u' : nV++; break;
            default  : nO++;
        }
    }
    printf("Vowels = %d and length = %d\n", nV, nV + nO);
}
```

Programming with Character Types

Ctype.h

- Defines number of functions on characters.
- `isascii()`, `isdigit()`, `isalpha()`, `isalnum()`, `isupper`, `islower()`, `isxdigit()`, `ispunct()`, `toupper()`, `tolower()`, etc.

Programming with Character Types

Example

```
#include <ctype.h>
#include <stdio.h>
int main() {
    char ch;

    printf("\nValid upper case characters: \n");
    for (ch = 0; isascii(ch); ch++)
        if (isupper(ch))
            printf("%c", tolower(ch)):
    printf("\n");

    printf("\nValid lower case characters: \n");
    for (ch = 0; isascii(ch); ch++)
        if (islower(ch))
            printf("%c", toupper(ch));
    printf("\n");
}
```

Programming with Character Types

Atoi

- Given an input consisting of digits read as characters.
- The program converts the input string to a number.
- The string may additionally include a negative sign for negative numbers.

Programming with Character Types

```c
#include <stdio.h>
int main() {
    char ch;
    int n = 0, sign = 1;

    printf("Enter a string of digits: ");
    ch = getchar();         // Read first character
    if (ch == '-') {
        sign = -1;          // Handle negative sign
        ch = getchar(); // Read next character
    }

    do {
        n = n * 10 + ch - 48; // Coversion formula
    } while((ch = getchar()) != '\n')

    printf("n = %d \n", n * sign);
}
```

Programming with Character Types

Counting Words

- Given a input text find the number of words in it.
- A word is separated from another word by at least one white space.
- Any number of white spaces may precede a word.
- End of the text is signalled by EOF character.
- EOF character is generated by CTRL-d.