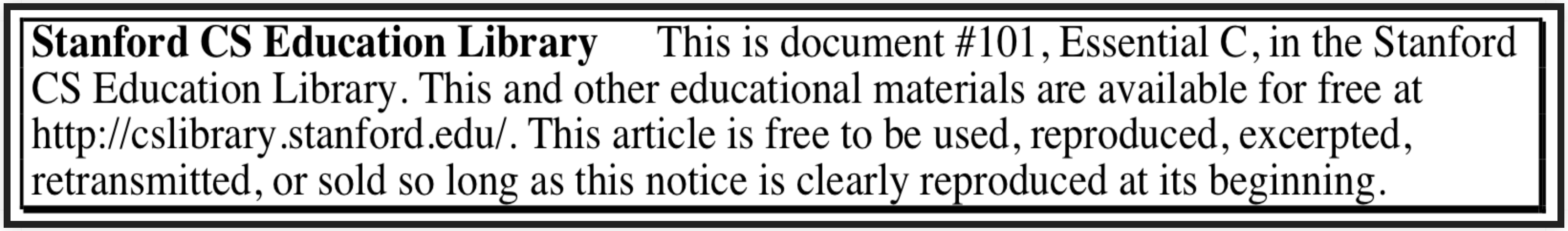
Miscellaneous

Stat 580: Statistical Computing

- Theme: Black - White
- Printable version

References

- Part of this slide set is based on *Essential C* by Nick Parlante:



Stanford CS Education Library This is document #101, Essential C, in the Stanford CS Education Library. This and other educational materials are available for free at http://cslibrary.stanford.edu/. This article is free to be used, reproduced, excerpted, retransmitted, or sold so long as this notice is clearly reproduced at its beginning.

(for copyright reason, this notice is reproduced here.)

Structures

Structures

- declaration such as `struct xxx` define a new type

```c
struct fraction {
int numerator;
int denominator;
};
```

- C uses the period (`.`) to access the fields in a record

```c
struct fraction f1;
f1.numerator = 1;
```

- you can copy two records of the same type using a single assignment statement
- however == does not work on structs
- C allows the use of array of structures

Example

```
#include<stdio.h>

struct fraction {
  int numerator;
  int denominator;
};

int main() {
  struct fraction f1, f2;          /* declare two fractions */
  f1.numerator = 22;
  f1.denominator = 7;
  f2 = f1;    /* this copies over the whole struct */
  return 0;
}
```

Example

```c
#include<stdio.h>

struct card {
  int num;
  char *suit;
};
void printcard(struct card c);

int main() {
  struct card c = {1, "diamond"};
  struct card *cp;

  printcard(c);
  cp = &c;
  printcard(*cp);

  /* (*struct_ptr).member is the same as struct_ptr->member */
  printf("%s %d\n", cp->suit, cp->num);
  /* what is *(cp->suit+2) */

  return 0;
}

void printcard(struct card c) {
  printf("%s %d\n", c.suit, c.num);
}
```

Example

```
#include<stdio.h>

struct card {
  int num;
  char *suit;
} cards[52]; /* extern storage class */

void create_cards();
void print_card(struct card c);

int main() {
  create_cards();
  print_card(cards[2]);
  return 0;
}
```

Example

```c
void create_cards(){
  int i, ind;

  for (i=0; i<52; i++) {
    cards[i].num = i % 13 + 1;
    ind = i / 13; /* integer division */
    switch(ind){
      case 0:
        cards[i].suit = "heart";
        break;
      case 1:
        cards[i].suit = "diamond";
        break;
      case 2:
        cards[i].suit = "spades";
        break;
      case 3:
        cards[i].suit = "clubs";
    }
  }
}
```

Command line arguments

Command line arguments

- allow user to specify arguments on the command line
- these arguments are handled using main() function arguments

```
int main(int argc, char *argv[])
```

  - `argc` is the number of arguments on the command line
    - including the name of the program itself
  - `argv` is an array of pointers to the values of the arguments
    - `argv[0]` is the name of the program
    - memory of these values are provided by the operating system
    - these values are treated as string and should be converted (via, e.g. `atoi`, `atof` from C standard library) if a numeric value is desired.

Example

```
#include <stdio.h>

int main(int argc, char *argv[]){
  int i;
  printf("Number of arguments: %d\n", argc);
  printf("Arguments:\n");
  for (i=0; i<argc; i++){
    printf("%s\n", argv[i]);
  }
  return 0;
}
```

Example

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]){
  char word[100];
  int n, i;

  if (argc != 3){
    printf("This program prints a word for n times\n");
    printf("usage: funname word n\n");
    printf("       word: the word that you want to repeat\n");
    printf("       n: number of times\n");
    return 1;
  }

  strcpy(word, argv[1]);
  n = atoi(argv[2]);

  for (i=0; i<n; i++)
    printf("%s\n", word);

  return 0;
}
```

File input and output

File

- data are usually stored in a file on your hard drive
- typical steps:

1. Inclusion of header file: `include <stdio.h>`
2. Declaration: declare a variable of (pointer) type FILE *
3. Connection: establish a connection between the variable and the file on your hard drive (`fopen()`)
4. I/O: perform I/O (`fgetc()`, `fscanf()`, `fputc()`, `fprintf()`)
5. Disconnection: break the connection (`fclose()`)

Example

```
#include <stdio.h>

int main(){
  FILE *f;     /* declaration */
  int x, y;

  f = fopen("abc.txt", "r"); /* connection */

  while(fscanf(f, "%d %d\n", &x, &y)==2)
    printf("%d %d\n", x, y);

  fclose(f);
  return 0;
}
```

abc.txt:

```
1 4
6 7
2 5
23 56
23 45
```

Example

```c
#include <stdio.h>

int main(){
  FILE *f;
  double slen, swid, plen, pwid;
  char species[20];

  f = fopen("iris.csv", "r");

  fscanf(f, "%*[^\n]\n", NULL);
  while(fscanf(f, "%lf,%lf,%lf,%lf,\"%[^\"]\"\n",
               &slen, &swid, &plen, &pwid, species)==5)
    printf("%f, %f, %f, %f, %s\n", slen, swid, plen, pwid, species);
  fclose(f);

  return 0;
}
```

First 5 rows of iris.csv:

```
"Sepal.Length","Sepal.Width","Petal.Length","Petal.Width","Species"
5.1,3.5,1.4,0.2,"setosa"
4.9,3,1.4,0.2,"setosa"
4.7,3.2,1.3,0.2,"setosa"
4.6,3.1,1.5,0.2,"setosa"
```