The C Info Sheet

C Programs

```
/*
 * Program name and purpose
 */
#include <stdio.h>
#include other libraries ...

#define CONST constant-expression

typedef type-expression Type; ...

type-expression globalVar; ...

int main(void)
{
   main program variables;

   main program code;

   return 0;
}

/*
 * Function name and purpose
 * Pre- and post- conditions
 */
return-type funcName(parameters ...)
{
   local variables;

   function code;
}
```

C Data

Basic data types:

```
int, long      - integer numbers
                 (e.g. 1, -1, -27, 1000)
float, double  - real numbers
                 (e.g. 1.0, 1e23, -0.005)
char           - characters
                 (e.g. 'a', 'b', '2', '!')
char *         - character strings
                 (e.g. "a string", "John")

typedef {FALSE, TRUE} Bool;
```

Structured data types:

`type a[N];`
  Array `a` of N objects of type *type*
  Indexed via `0 .. N-1`

`struct {t1 a; t2 b; t3 c} d;`
  Record `d` with fields `(a, b, c)` of types `(t1, t2, t3)`

C Expressions

Arithmetic operators:
`+  -  *  /  %` (modulus)
Note: no exponentiation operator like `^` in Miranda

Relational operators:
`==  !=  <  >  <=  >=`

Logical operators:
`!` (NOT) `&&` (AND) `||` (OR)

Bitwise operators:
`~` (NOT) `&` (AND) `|` (OR) `^` (XOR)

C Statements

Assignment:

```
var = expression;
x++;  ++x;  x+=expr;  x-=expr; ...
```

`if` Statement:

```
if (condition1) { statements1; }
else if (condition2) { statements2; }
...
else if (conditionn) { statementsn; }
else { statementsn+1; }
```

`switch` Statement:

```
switch (expression) {
case const1: statements1; break;
case const2: statements2; break;
...
case constn: statementsn; break;
default: statementsn+1;
}
```

`while` Statement:

```
while (condition) { statements; }

break - exit the loop
continue - go to next iteration
```

`do` Statement:

```
do { statements; } while (condition);
```

`for` Statement:

```
for (init; condition; next) { statements; }
```

which is equivalent to

```
init;  while (condition) { statements; next; }
```

C Idioms

Read in text input, character by character:

```
int c;
while ((c = getchar()) != EOF) {
   process c;
}
```

Read in text input, line by line:

```
char s[N];
while (gets(s) != NULL) {
   process s;
}
```

Read in all integer value input:

```
int i;
while (scanf("%d", &i) != EOF) {
    process i;
}
```

Scan along an array:

```
int i, a[N];
for (i = 0; i < N; ++i) {
   process a[i];
}
```

Scan along a linked-list:

```
struct n {int data; struct n *next;};
typedef struct n Node;
typedef Node *List;
List L;  Node *p;
for (p = L; p!= NULL;  p = p->next) {
   process p->data;
}
```

`<stdio.h>` Library

`FILE *f;` - external file handle

`int getchar(void), getc(FILE *f);`
Fetch one character from input

`int putchar(int ch), putc(int ch, FILE *f);`
Display a character on output

`char *gets(char *s);`
`char *fgets(char *s, int len, FILE *f);`
Fetch one line into buffer

`scanf(`*`format-string`*`,` *`address`*$_1$`, ...);`
Interpret input according to formats;
store results in specified addresses;
return number of valid, stored values
e.g.
`scanf("%d", &i);` - read an integer
`scanf("%f", &r);` - read a real number
`scanf("%c", &c);` - read a character
`scanf("%s", s);` - read a string

`printf(`*`format-string`*`,` *`expr`*$_1$`, ...);`
Display expressions according to specified
formats; expression type must match format
e.g.
`printf("%3d", i);` - show an integer
`printf("%.2f", r);` - show a real number
`printf("%c", c);` - show a character
`printf("%s", s);` - show a string
`printf("Hello\n");` - show literal text

`<stdlib.h>` Library

`EOF` - end of file marker for i/o
`NULL` - canonical invalid pointer

`void *malloc(int size);`
allocate memory and return address

`void free(void *addr);`
release previously allocated memory

`int rand(void);`
generate a random `int` value

`void exit(int status);`
terminate program with return status

`<ctype.h>` Library

`Bool isalpha(char), isdigit(char), isspace(char),`
`ispunct(char), islower(char), isupper(char);`
character classification functions

`char toupper(char), tolower(char);`
case conversion functions

`<math.h>` Library

`double sin(double), cos(double), tan(double);`
trigonometric functions - inputs are radians

`double log(double), pow(double, double);`
`double sqrt(double);`
logarithms, raise to power, square root

`<string.h>` Library

`char *strcpy(char *buff, char *src);`
Copy `src` string int `buffer`

`int strcmp(char *s1, char *s2);`
Compare two strings; return difference

`int strlen(char *s)`
Return length of string `s`