CSI 402 – Systems Programming

A Complete Program with Header Files

Handout 1.2

The following simple program uses several header files and two source files. The contents of all the files are shown below.

File: `constants.h`

```
#define  MINKEY   1
#define  MAXKEY   100
```

File: `struct_def.h`

```
struct  key_record {
   int  value;
   struct key_record *next;
};

typedef  struct key_record*  keyptr;
```

File: `globals.h`

```
keyptr  h;  /* Pointer to first node of list. */
```

File: `externs.h`

```
extern  keyptr  h; /* Pointer to first node of list (external). */
```

File: `prototypes.h`

```
void   insert_key(int);
void   print_list(void);
```

File: `main.c`

```
#include  <stdio.h>
#include  "constants.h"
#include  "struct_def.h"
#include  "globals.h"
#include  "prototypes.h"

int  main(void) {

  int   key;
  h = NULL; /* Initialize list to empty. */

  /* Repeatedly obtain keys from the user. Insert a key into the list */
  /* if it is valid (i.e., it is in the range MINKEY through MAXKEY). */
  /* When an invalid key is given, print the list and stop.           */
```

File: `main.c` (continued)

```
  while (1) {
      printf("Key value? "); scanf("%d", &key);
      if ((key < MINKEY) || (key > MAXKEY)) {
         /* Invalid key. */
         print_list(); break;
      }
      else insert_key(key);
  }  /* End of while. */

  return 0;

} /* End of main. */
```

File: `funct.c`

```
#include  <stdio.h>
/* The <stdlib.h> header file is needed for malloc. */
#include  <stdlib.h>
#include  "constants.h"
#include  "struct_def.h"
#include  "externs.h"

void insert_key (int k) {

  /* Inserts key given by parameter k into the list. */
  /* (Assumes that key is valid.)                     */

  keyptr  x, cur, prev;

  if ((x = (keyptr) malloc(sizeof(struct key_record))) == NULL) {
     printf("Allocation failed.\n"); exit(1);
  }

  /* Obtained space for a new key record. */
  x->value = k;  x->next = NULL;
  if (h == NULL) { /* List is currently empty. */
     h = x;
  }
  else {
     /* Move to the last node of the list and then insert. */
     cur = h;  prev = NULL;
     while (cur != NULL) {
       prev = cur;  cur = cur->next;
     }
     prev->next = x;
  }

} /* End of insert_key. */
```

File: `funct.c` (continued)

```
void print_list (void) {

   /* Prints the list pointed to by the global variable h. */

   keyptr  cur;
   if (h == NULL)
      printf("The list is empty.\n");
   else {
      cur = h;
      while (cur != NULL) {
         printf("%d\n", cur->value); cur = cur->next;
      }
   }

} /* End of print_list. */
```

To generate the executable version (`a.out`) of the above program, you would use the following sequence of commands:

```
    gcc  -c  main.c
    gcc  -c  funct.c
    gcc  main.o  funct.o
```

The use of `make` will simplify the process of selectively recompiling modified files and generating the executable version of a program in multiple files.