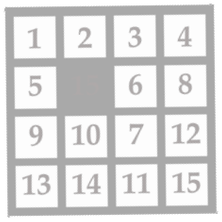
This is CS 50.



Harvard College's Introduction to Computer Science I

COMPUTER SCIENCE 50

WEEK 4

DAVID J. MALAN '99
malan@post.harvard.edu

Passing by Value

```
void
swap(int a, int b)
{
    int tmp;

    tmp = a;
    a = b;
    b = tmp;
}
```

see
buggy3.c

Passing by Pointer

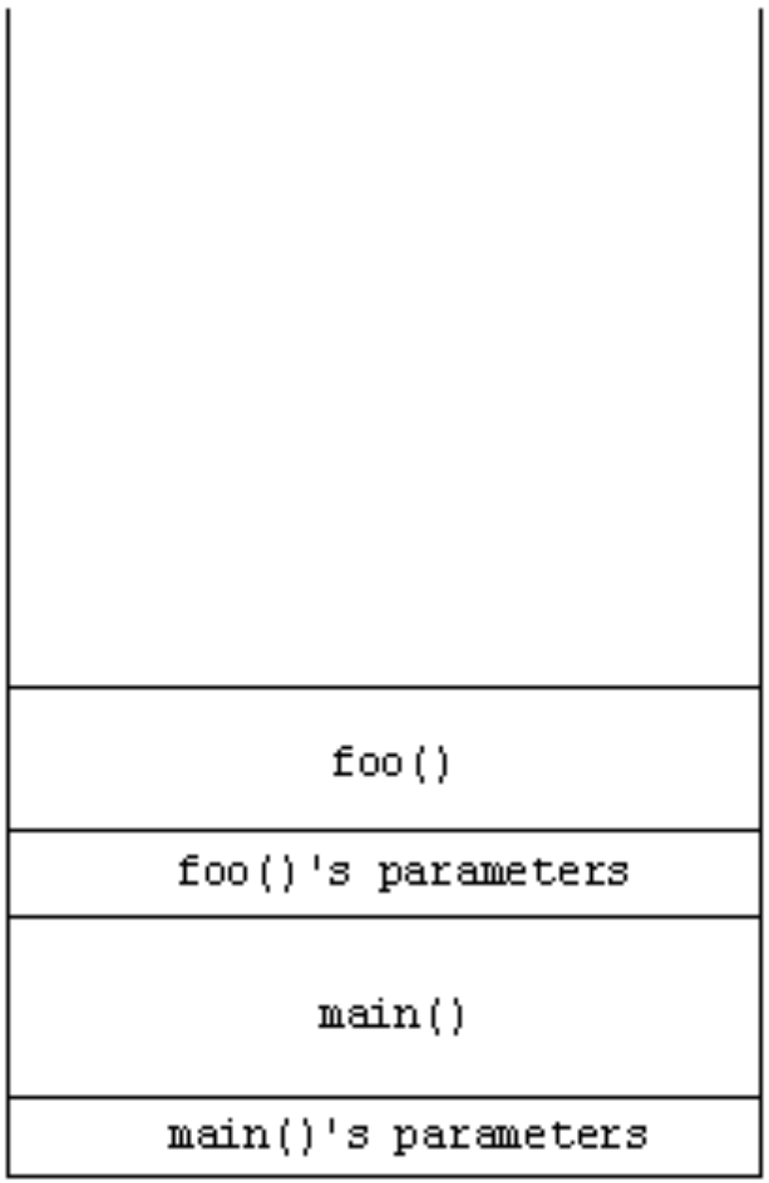
```
void
swap(int *a, int *b)
{
    int tmp;

    tmp = *a;
    *a = *b;
    *b = tmp;
}
```

see
swap.c

The Stack

Revisited



Pointers

```
int i, j;
int *p;
```

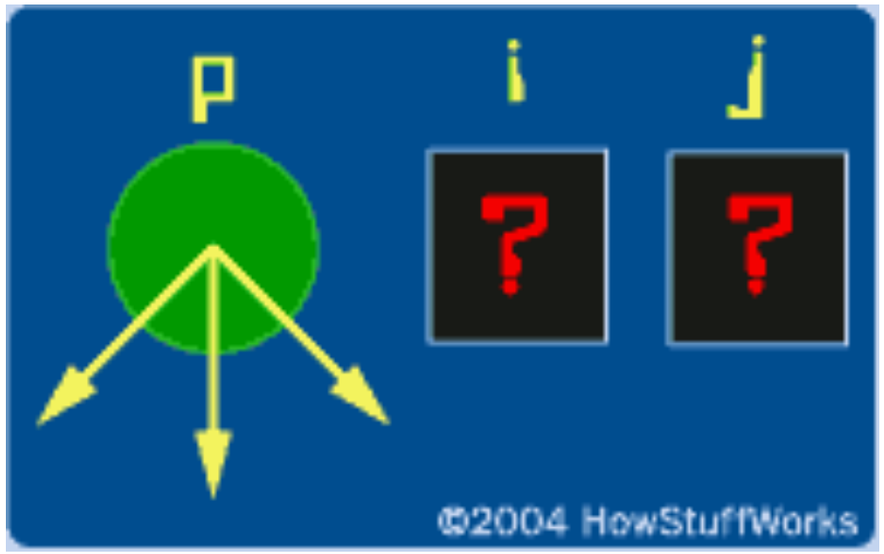


Image from http://computer.howstuffworks.com/c22.htm.

Pointers

```
p = &i;
```

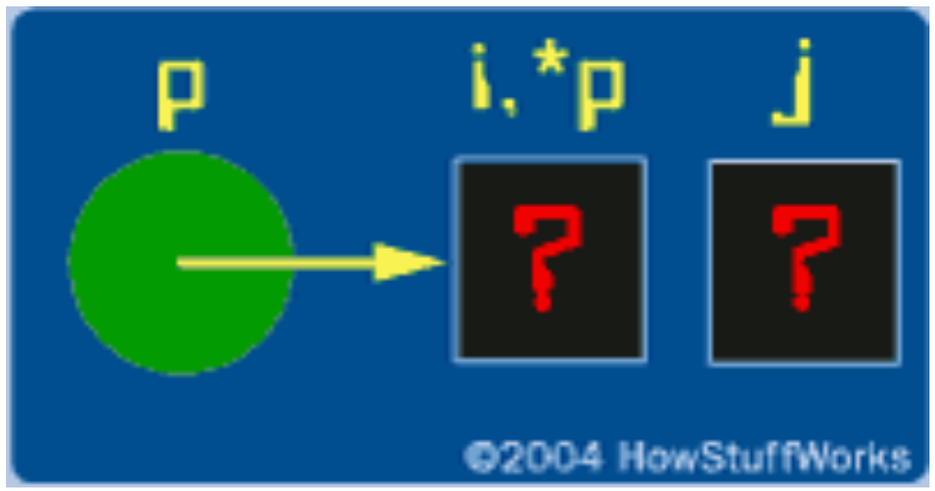


Image from http://computer.howstuffworks.com/c22.htm.

Pointers

```
*p = 5;
```

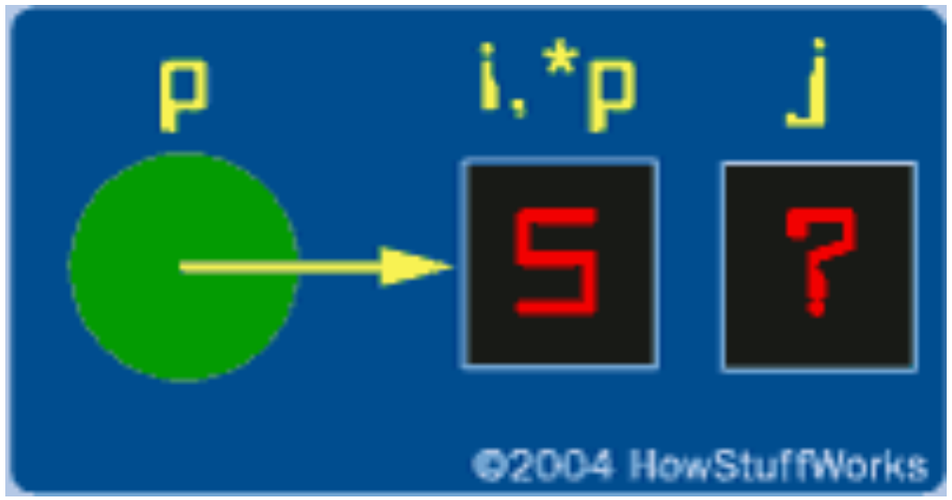
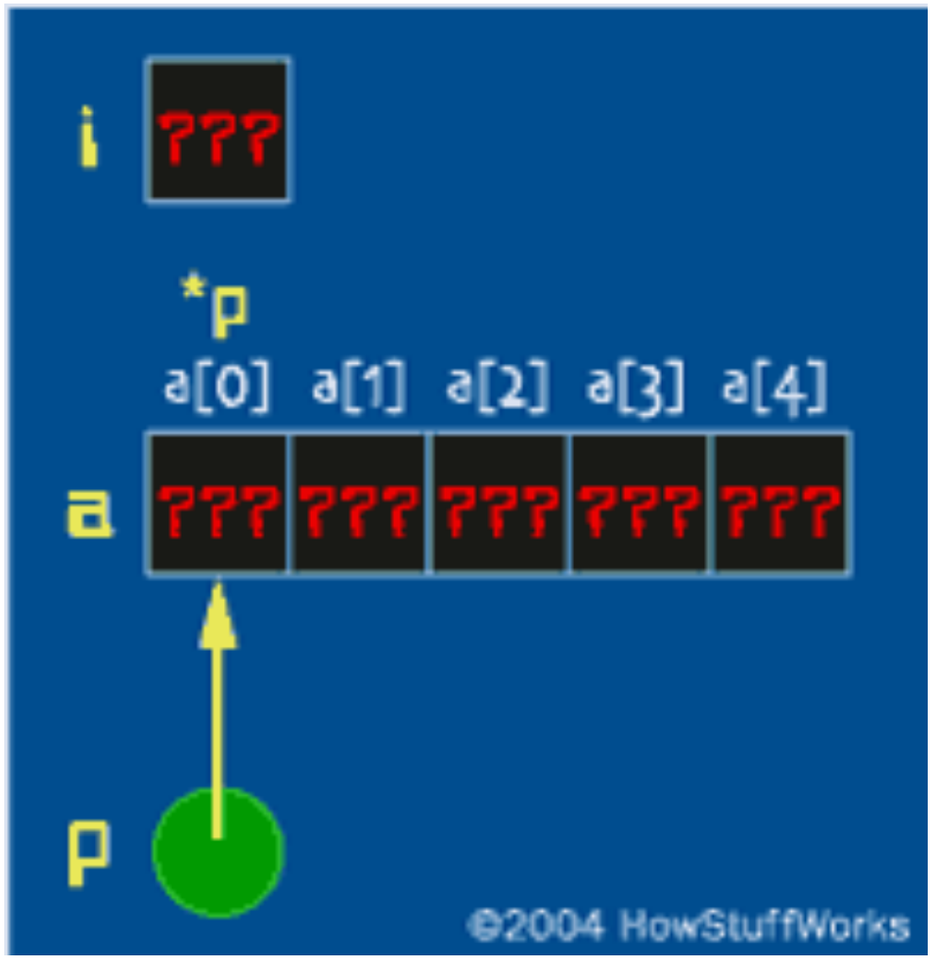


Image from http://computer.howstuffworks.com/c22.htm.

Arrays

```
int i;
int a[5];
int *p = a;
```



see
compare{1,2}.c, pointers{1,2}.c

Image from http://computer.howstuffworks.com/c22.htm.

Dynamic Memory Allocation

malloc

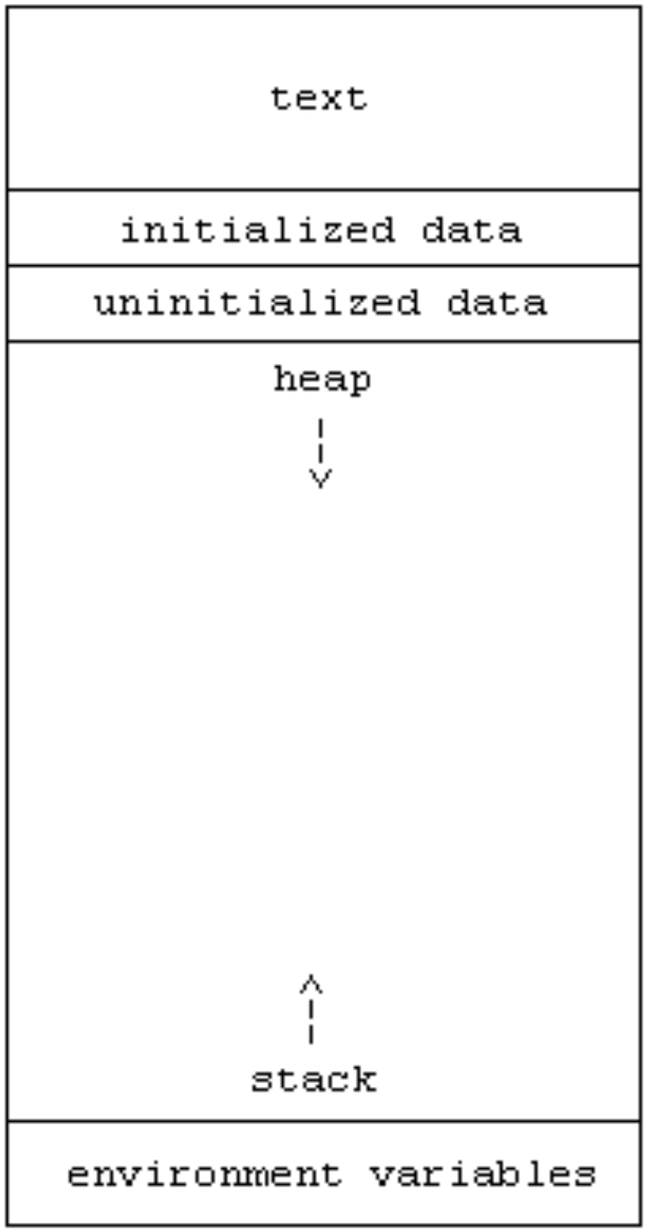
```
// get line of text
printf("Say something: ");
char *s1 = GetString();
if (s1 == NULL)
    return 1;

// allocate enough space for copy
char *s2 = malloc(strlen(s1) * sizeof(char) + 1);
if (s2 == NULL)
    return 1;
```

see
copy{1,2}.c

Memory Management

Revisited



CS 50's Library

Revisited

```
bool

string

char GetChar();

double GetDouble();

float GetFloat();

int GetInt();

long long GetLongLong();

string GetString();
```

see
scanf{1,2,3}.c, http://cs50.net/pub/releases/cs50/cs50.{c,h}

Dangerous Functions

```
gets → fgetc
scanf → fgetc


strcpy → strncpy
strcat → strncat


printf
fprintf


...
```

Safe Code

```
int
main(int argc, char *argv[])
{
    int i;
    for (i = 1; i < argc; i++) {
        printf("%s ", argv[i]);
    }
    printf("\n");
}
```

Unsafe Code

```
void echo_arg(const char s[])
{
    char buf[MAX_BUF_SIZE];
    strcpy(buf, s);
    printf("%s ", buf);
}

int main(int argc, char *argv[])
{
    int i;
    for (i = 1; i < argc; i++) {
        echo_arg(argv[i]);
    }
    printf("\n");
}
```

Unsafe Code (2)

```
void gotcha()
{
    printf("\nGotcha!\n");
}

void echo_arg(const char s[])
{
    char buf[MAX_BUF_SIZE];
    strcpy(buf, s);
    printf("%s ", buf);
}

int main(int argc, char *argv[])
{
    int i;
    for (i = 1; i < argc; i++) {
        echo_arg(argv[i]);
    }
    printf("\n");
}
```

struct

(and header files)

```
typedef struct
{
    int id;
    char *name;
    char *house;
}
student;
```

see
structs.h, structs2.c

File I/O

```
fopen/fclose

fscanf/fprintf

fread/fwrite

feof

...
```

• • •

see
structs2.c