sizeof()

- Compile-time operator*
- Compute the size of operand in **bytes** (8 bits = 1 byte)
  - Can compute size of **type** or **variable**
- May give different output depending on machine!

sizeof()

```
sizeof(int)     // == 4 (bytes -- 32 bits)

int A[] = {1, 2, 3, 4, 5};
sizeof(A)       // == ?
```

sizeof()

```
sizeof(int)     // == 4 (bytes -- 32 bits)

int A[] = {1, 2, 3, 4, 5};
sizeof(A)         // ==  20
sizeof(A) / sizeof(int)
                // == 5
```

Warning: this does **not** work for arrays passed to functions. It only works where the array was defined.

sizeof()

```
sizeof(char)

sizeof(double)

sizeof(float)

sizeof(int *)

sizeof(char *)
```

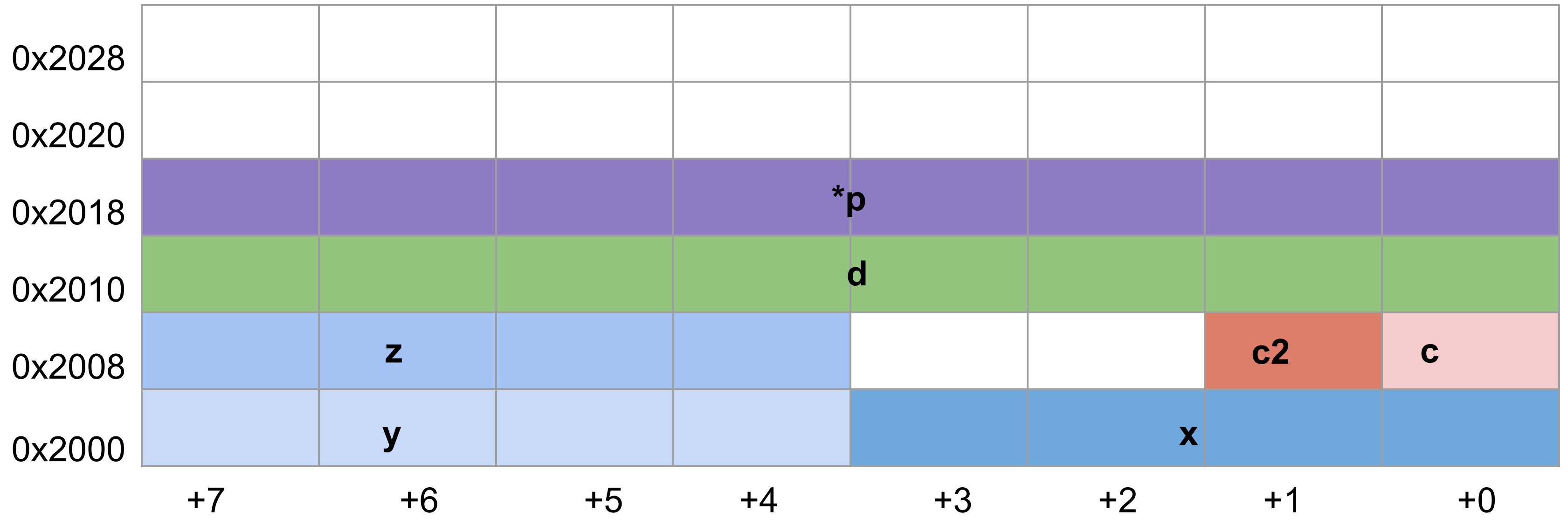
sizeof()

```
sizeof(char)        // 1

sizeof(double)      // 8

sizeof(float)       // 4

sizeof(int *)       // 32bit/grok: 4,  64bit: 8

sizeof(char *)      // 32bit/grok: 4,  64bit: 8
```
`sizeof(<type>*)     // 32bit/grok: 4,  64bit: 8`

```
int x;
char c;
int y;
double d;
char c2;
int z;
int *p = &x;
```

char (1B) int (4B)
double (8B) int * (8B)

Each cell represents 1 byte



sizeof() in functions

```
int A[] = {1, 2, 3, 4};
printf(“%d\n”, sizeof(A)); // ?
double_nums(A, 4);
…

void double_nums(int A[], int n) {
   …
   printf(“%d\n”, sizeof(A)); // ?
}
```

sizeof() in functions

```
int A[] = {1, 2, 3, 4};
printf(“%d\n”, sizeof(A)); // ?
double_nums(A, 4);
…

void double_nums(int A[], int n) {
   …
   printf(“%d\n”, sizeof(A)); // (program.c:12)
}
```

```
program.c:12:23: error: sizeof on array function parameter will return
size of 'int *' instead of 'int []' [-Werror,-Wsizeof-array-argument]
```

sizeof() in functions

```
int A[] = {1, 2, 3, 4};
printf("%d\n", sizeof(A)); // 16
double_nums(A, 4);
…

void double_nums(int A[], int n) {
   …
   printf("%d\n", sizeof((int *) A)); // 4
}
```

Can't get the sizeof arrays passed to functions!

sizeof structs (W8)

sizeof structs

```
struct {
    int id;
    char name[21];
    int year_opened;
    double balance;
} account;

sizeof(account.name) ?
sizeof(account) ?
```

sizeof structs

```
struct {
    int id;
    char name[21];
    int year_opened;
    double balance;
} account;

sizeof(account.name) = sizeof(char) * 21 = 1 * 21 = 21
sizeof(account) ?
```

sizeof structs

```
struct {
    int id;
    char name[21];
    int year_opened;
    double balance;
} account;

sizeof(account.name) = sizeof(char) * 21 = 1 * 21 = 21
sizeof(account)
  = sizeof(account.id) + sizeof(account.name)
    + sizeof(account.year_opened) + sizeof(account.balance)
  = sizeof(int) + 21 + sizeof(int) + sizeof(double)
  = 4 + 21 + 4 + 8 = 37  (bytes)
```

sizeof structs

```
struct {
    int id;
    char *name;
    int year_opened;
    double balance;
} account;
account.name = “Hello”;

sizeof(account.name) = ?
sizeof(account) = ?
```

sizeof structs

```
struct {
    int id;
    char *name;
    int year_opened;
    double balance;
} account;
account.name = “Hello”;

sizeof(account.name) = sizeof(char *) = 8 (64-bit), 4 (32-bit)
sizeof(account) = ?
```

sizeof structs

```
struct {
    int id;
    char *name;
    int year_opened;
    double balance;
} account;
account.name = “Hello”;

sizeof(account.name) = sizeof(char *) = 8 (64-bit), 4 (32-bit)
sizeof(account) = 4 + 8 + 4 + 8 = 24

That’s ok, because this is a string literal.
```

sizeof structs

```
struct {
    int id;
    char *name;
    int year_opened;
    double balance;
} account;


char name[] = “Namey name”; // assume from input.
account.name = malloc(strlen(name) + 1); assert(account.name);
strcpy(account.name, name);

sizeof(account) = 4 + 8 + 4 + 8 = 24    - unchanged
Now there’s 11 bytes (sizeof(name)) unaccounted for! Careful!
```

sizeof structs

```
struct {
    int id;
    char *name;
    size_t name_size;  // if we need the size.
    int year_opened;
    double balance;
} account;

char name[] = “Namey name”; // assume from input.
account.name_size = strlen(name) + 1;
account.name = malloc(account.name_size);  assert(account.name);
strcpy(account.name, name);
sizeof(account) = 4 + 8 + 4 + 4 + 8 = 28  (4B to store name_size)
name_size + sizeof(account) = 28 + 11 = 41  <- actual size of struct.
```