C

CS 045

Computer Organization and Architecture

Prof. Donald J. Patterson

TYPES IN C

NUMBERS

- ints
  - size not guaranteed

```c
#include <stdio.h>
#include <limits.h>

int main(){

    char a = CHAR_MAX;
    unsigned char b = UCHAR_MAX;

    short c = SHRT_MAX;
    unsigned short d = USHRT_MAX ;

    int e = INT_MAX;
    unsigned int f = UINT_MAX;

    long g = LONG_MAX;
    unsigned long h = ULONG_MAX;

    long long i = LLONG_MAX;
    unsigned long long j = ULLONG_MAX;

    printf("char               size: %lu, max: %d\n",sizeof(a),a);
    printf("unsigned char      size: %lu, max: %d\n",sizeof(b),b);

    printf("short              size: %lu, max: %d\n",sizeof(c),c);
    printf("unsigned short     size: %lu, max: %d\n",sizeof(d),d);

    printf("int                size: %lu, max: %d\n",sizeof(e),e);
    printf("unsigned int       size: %lu, max: %u\n",sizeof(f),f);

    printf("long               size: %lu, max: %ld\n",sizeof(g), g);
    printf("unsigned long      size: %lu, max: %lu\n",sizeof(h), h);

    printf("long long          size: %lu, max: %lli\n",sizeof(i), i);
    printf("unsigned long long size: %lu, max: %llu\n",sizeof(j), j);

    return 0;
}
```

TYPES IN C

NUMBERS

- ints
  - size not guaranteed

```
char                 size: 1, max: 127
unsigned char        size: 1, max: 255
short                size: 2, max: 32767
unsigned short       size: 2, max: 65535
int                  size: 4, max: 2147483647
unsigned int         size: 4, max: 4294967295
long                 size: 8, max: 9223372036854775807
unsigned long        size: 8, max: 18446744073709551615
long long            size: 8, max: 9223372036854775807
unsigned long long   size: 8, max: 18446744073709551615
```

```c
#include <stdio.h>
#include <limits.h>

int main(){

    char a = CHAR_MAX;
    unsigned char b = UCHAR_MAX;

    short c = SHRT_MAX;
    unsigned short d = USHRT_MAX ;

    int e = INT_MAX;
    unsigned int f = UINT_MAX;

    long g = LONG_MAX;
    unsigned long h = ULONG_MAX;

    long long i = LLONG_MAX;
    unsigned long long j = ULLONG_MAX;

    printf("char                 size: %lu, max: %d\n",sizeof(a),a);
    printf("unsigned char        size: %lu, max: %d\n",sizeof(b),b);

    printf("short                size: %lu, max: %d\n",sizeof(c),c);
    printf("unsigned short       size: %lu, max: %d\n",sizeof(d),d);

    printf("int                  size: %lu, max: %d\n",sizeof(e),e);
    printf("unsigned int         size: %lu, max: %u\n",sizeof(f),f);

    printf("long                 size: %lu, max: %ld\n",sizeof(g), g);
    printf("unsigned long        size: %lu, max: %lu\n",sizeof(h), h);

    printf("long long            size: %lu, max: %lli\n",sizeof(i), i);
    printf("unsigned long long   size: %lu, max: %llu\n",sizeof(j), j);

    return 0;
}
```

TYPES IN C

NUMBERS

- ints
  - size guaranteed

```c
#include <stdio.h>
#include <limits.h>
#include <inttypes.h>

int main(){

    int8_t a = INT8_MAX;
    uint8_t b = UINT8_MAX;

    int16_t c = INT16_MAX;
    uint16_t d = UINT16_MAX ;

    int32_t e = INT32_MAX;
    uint32_t f = UINT32_MAX;

    int64_t g = INT64_MAX;
    uint64_t h = UINT64_MAX;

    printf("int8_t   size: %lu, max: %d\n",sizeof(a),a);
    printf("uint8_t  size: %lu, max: %d\n",sizeof(b),b);

    printf("int16_t  size: %lu, max: %d\n",sizeof(c),c);
    printf("uint16_t size: %lu, max: %d\n",sizeof(d),d);

    printf("int32_t  size: %lu, max: %d\n",sizeof(e),e);
    printf("uint32_t size: %lu, max: %u\n",sizeof(f),f);

    printf("int64_t  size: %lu, max: %lld\n",sizeof(g), g);
    printf("uint64_t size: %lu, max: %llu\n",sizeof(h), h);

    return 0;
}
```

TYPES IN C

NUMBERS

- ints
  - size guaranteed

```
int8_t   size: 1, max: 127
uint8_t  size: 1, max: 255
int16_t  size: 2, max: 32767
uint16_t size: 2, max: 65535
int32_t  size: 4, max: 2147483647
uint32_t size: 4, max: 4294967295
int64_t  size: 8, max: 9223372036854775807
uint64_t size: 8, max: 18446744073709551615
```

```c
#include <stdio.h>
#include <limits.h>
#include <inttypes.h>

int main(){

    int8_t a = INT8_MAX;
    uint8_t b = UINT8_MAX;

    int16_t c = INT16_MAX;
    uint16_t d = UINT16_MAX ;

    int32_t e = INT32_MAX;
    uint32_t f = UINT32_MAX;

    int64_t g = INT64_MAX;
    uint64_t h = UINT64_MAX;

    printf("int8_t   size: %lu, max: %d\n",sizeof(a),a);
    printf("uint8_t  size: %lu, max: %d\n",sizeof(b),b);

    printf("int16_t  size: %lu, max: %d\n",sizeof(c),c);
    printf("uint16_t size: %lu, max: %d\n",sizeof(d),d);

    printf("int32_t  size: %lu, max: %d\n",sizeof(e),e);
    printf("uint32_t size: %lu, max: %u\n",sizeof(f),f);

    printf("int64_t  size: %lu, max: %lld\n",sizeof(g), g);
    printf("uint64_t size: %lu, max: %llu\n",sizeof(h), h);

    return 0;
}
```

TYPES IN C

NUMBERS

- ints
  - size guaranteed

Type category	Signed types			Unsigned types		
	Type	Minimum value	Maximum value	Type	Minimum value	Maximum value
Exact width	`intN_t`	`INTN_MIN`	`INTN_MAX`	`uintN_t`	0	`UINTN_MAX`
Least width	`int_leastN_t`	`INT_LEASTN_MIN`	`INT_LEASTN_MAX`	`uint_leastN_t`	0	`UINT_LEASTN_MAX`
Fastest	`int_fastN_t`	`INT_FASTN_MIN`	`INT_FASTN_MAX`	`uint_fastN_t`	0	`UINT_FASTN_MAX`
Pointer	`intptr_t`	`INTPTR_MIN`	`INTPTR_MAX`	`uintptr_t`	0	`UINTPTR_MAX`
Maximum width	`intmax_t`	`INTMAX_MIN`	`INTMAX_MAX`	`uintmax_t`	0	`UINTMAX_MAX`

TYPES IN C

NUMBERS

- floats

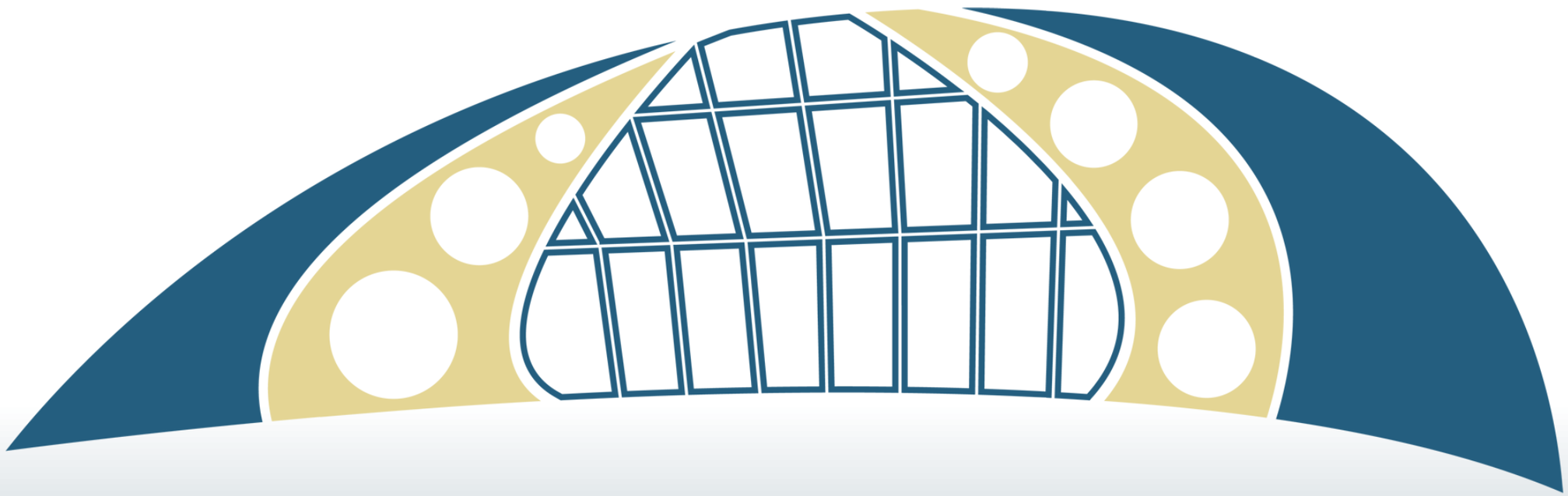
```c
#include <stdio.h>
#include <float.h>

int main(){

    float a = FLT_MAX;
    double b = DBL_MAX;
    long double c = LDBL_MAX;

    printf("float           size: %lu, max: %f\n",sizeof(a),a);
    printf("double          size: %lu, max: %lf\n",sizeof(b),b);
    printf("long double     size: %lu, max: %Lf\n",sizeof(c),c);

    return 0;
}
```

```
float          size: 4, max: 340282346638528859811704183484516925440.000000
double         size: 8, max: 179769313486231570814527423731704356798070567525844996598917476803157260780028538760589558632766878171540458953514382464234321326889464182768467546703537516986049910576551282076245490090389328944075868508455133942304583236903222948165808559332123348274797826204144723168738177180919299881250404026184124858368.000000
long double    size: 16, max: 11897314953572317650212638530309702051690633222946242004403237338917370055229707226164102903365288828535456978074955773144274431536702884341981255738537436786735932007069732632019159182829615243655295106467910866143117906321697788388961347865606003991487534332114549111600886798451548665128523401497730376000091254793939662231513836224178385427439178381387178058894875405751682263476592355769748051137256490208848552224947913993775850260117735491800997962260268595085588836081598469002356451323465944763849398592764562845796617729304078066092291027150460853880879593277816229868275478307680800401506949423034117289577771003357140105597752421240573470073862516601108283791196230084692772009651535002084744707924438485459128867230006190851264721119513614675276335195629275979572502780029807959041931396030214709970352764674455309220226796562809914982320833296412410385092391847347861219216972105434842870483534081130425730022164213489173471742348007148807510020643905172342476560047217680964861079949434157034763206435586242074435044243805661360176088374781653890278095769759772868600714870282879555671414046326158326236027628963161739784842544868606099482708679680480787025118589308385465842230409088059962945945862019037660484467909260022254105307759010657606713472001258464069570302571389609837579989269545530523685607586831792231136395194688508807718721047052039575874800131431314442549439199401757531693393923668818561891299317291042529212368351599223220509980016771027840353601408292963981151228777681357060457893435354516965395612540488464471697868932116710872290880827783505182288576460622187397028516550837209923494833344352289847512327537266360662139022812647062340753520717240586650795182173034637826313533937067749019501978416904418247380631628285868577414325811653640402184027249133933209492194984224427304270198730445366203502623869578046820036014472919971230955300572061418669748528468561865148327159744812031219467516863793430961896151073300655524214851952017628585950910518394725028638716324941676138049963197914418702543027067584951920088379151694015817400467114778772014596444611752040594535047647218079757611117208462736392796003396704700376133745095531841500737964126050479232516613548412918842113408230154733047540670728187635036173329080059518963252070716739045477771296822652062256514399193768044002923809031124379126147762559646942219813751469670794468703580043925076594516183798118593920495440361149153107822510726914869798092409467721427270124043771874092167566136349389004512323516681460893224006979931760178053381918499819330084109859939387602926013909114145260037202848721324119554242821018312042161044674046216353369005836646065911562987647455250681450039329414041314954006776029510059622530228230036314738246810596484424413248645731374375950964161680480241293518762046681356368775328146755387988717718365128939471953350618850032676073543886733680020743878496570145760903498575712430451020387304948542567024793393228091105260415385289948492039910919461299124916332899179980943803378795220931314669461497059396641523759492858909604899161219449899863848370224866722491489246784102061833646274169695763076324802355879752452537370354338829608627534277400163334340550835370485073745448197547222289752810830208986826330202852599230841680545396879114182976299889645764827652875045628549242651652177507995162596692291149777889623566709566271384820181913483216879958636526376209782850700993372943967846398790249145142227425270063639423279984839767399871544185542015622441549266530145155046854892586202760857618371297633587612153825651296335381416639495165560002641591865548500570526114319529199188079545223946496276356301785808966922640623538289853586759599064700838568712381032959192649484625076899225841930548076362021508902214922052806984201835084058693849381549890944546197789302911357651677540623227829831403347327660395223160342282471752818181884430488092132193355086987339586127607367086665237555567580317149010847732009642431878007000879734603290627894355374356444885190719161645514115576193939969076741515640282654366402676009508752394550734155613586793306603174472092444651353236664764973540085196704077110364053815007348689179836404957060618953500508984091382686953509006678332447257871219660441528492484004185093281190896363417573989716659600075948780061916409485433875852065711654107226099628815012314437794400874930194474433078438899570184271000480830501217712356062289507626904285680004771889315808935851559386317665294808903126774702966254511086154895839508779675546413794489596052797520987481383976257859210575628440175934932416214833956535018919681138909184379573470326940634289008780584694035245347939808067427323629788710086717580253156130235606487870925986528841635097252953709111431720488774740553905400942537542411931794417513706468964386151771884986701034153254238591108962471088538580868883777725864856414593426212108664758848926003176234596076950884914966244415660441955208681198977024 0.000000
```

TYPES IN C

BOOLEAN

- Traditionally there is no boolean type in C
  - use a char set to 0 or 1
    - 0 = false
    - anything else (e.g. 1) = true



WESTMONT INSPIRED

COMPUTING LAB