Fundamentals of C

First C program – print on screen

```
#include <stdio.h>
int main()
{
    printf ("Hello, World! \n") ;
    return 0;
}
```

Output

```
Hello, World!
```

More print

```
#include <stdio.h>
int main()
{
    printf ("Hello, World! ") ;
    printf ("Hello \n World! \n") ;
    return 0;
}
```

Output

```
Hello, World! Hello
 World!
```

A Simple C program

```
#include <stdio.h>
int main()
{
    int x, y, sum, max;
    scanf("%d%d", &x, &y);
    sum = x + y;
    if (x > y) max = x;
    else  max = y;
     printf ("Sum = %d\n",  sum);
     printf ("Larger = %d\n",  max);
     return 0;
}
```

When you run the program

Output after you type 15 and 20

```
15 20
Sum = 35
Larger = 20
```

Reading values from keyboard

```
#include <stdio.h>
int main()
{
        int num ;
        scanf ("%d", &num) ;
        printf (“No. of students is %d\n”, num);
        return 0;
}
```

Centigrade to Fahrenheit

```
#include <stdio.h>
int main()
{
    float cent, fahr;
    scanf("%f",&cent);
    fahr = cent*(9.0/5.0) + 32;
    printf( "%f C equals %f F\n", cent, fahr);
    return 0;
}
```

What does this do?

```
#include <stdio.h>
int main()
{
    int x, y;
    scanf(“%d%d”,&x,&y);
    if (x>y) printf(“Largest is %d\n”,x);
    printf(“Largest is %d\n”,y);
    return 0;
}
```

Structure of a C program

- A collection of functions (we will see what they are later)
- Exactly one special function named main must be present. Program always starts from there
- Each function has statements (instructions) for declaration, assignment, condition check, looping etc.
- Statements are executed one by one

```
#include <stdio.h>
int main()
{
    int x, y, sum, max;
    scanf("%d%d", &x, &y);
    sum = x + y;
    if (x > y)
      max = x;
    else
      max = y;
    printf ("Sum = %d\n", sum);
    printf ("Larger = %d\n", max);
    return 0;
}
```

main function

Declaration statement

Input statement

Assignment statements

Control statement

Output statement

Return statement

Writing a C program

- You will have to understand what different statements do to decide which you should use in what order to solve your problem
- There is a fixed format (“syntax”) for writing each statement and other things. Need to remember the syntax
  - Do not question why you have to type exactly like this, you just have to or it is not a C program!!
  - Compiler will give error if your typed program does not match required C syntax
- There are other rules to follow

Things you will see in a C program (we will look at all these one by one)

- Variables
- Constants
- Expressions (Arithmetic, Logical, Assignment)
- Statements (Declaration, Assignment, Control (Conditional/Branching, Looping))
- Arrays
- Functions
- Structures
- Pointers
- Few other things….

The C Character Set

- The C language alphabet
  - Uppercase letters ‘A’ to ‘Z’
  - Lowercase letters ‘a’ to ‘z’
  - Digits ‘0’ to ‘9’
  - Certain special characters:

!	**#**	**%**	**^**	**&**	*****	**(**	**)**
-	**_**	**+**	**=**	**~**	**[**	**]**	****
|	**;**	**:**	**‘**	**“**	**{**	**}**	**,**
.	**<**	**>**	**/**	**?**	**blank**		

A C program should not contain anything else

Variables

- Very important concept for programming
- An entity that has a value and is known to the program by a name
- Can store any temporary result while executing a program
- Can have only one value assigned to it at any given time during the execution of the program
- The value of a variable can be changed during the execution of the program

Contd.

- Variables stored in memory
- Remember that memory is a list of storage locations, each having a unique address
- A variable is like a bin
  - The contents of the bin is the value of the variable
  - The variable name is used to refer to the value of the variable
  - A variable is mapped to a location of the memory, called its address

Example

```
#include <stdio.h>
int main( )
{
        int X;
        int Y;
        X=10;  X=20;  X=X+1;  X=X*5;
        Y=15;  Y=Y+3;  Y=Y/6;
        printf(“X = %d, Y= %d\n", X, Y);
         return 0;
}
```

Variables in Memory

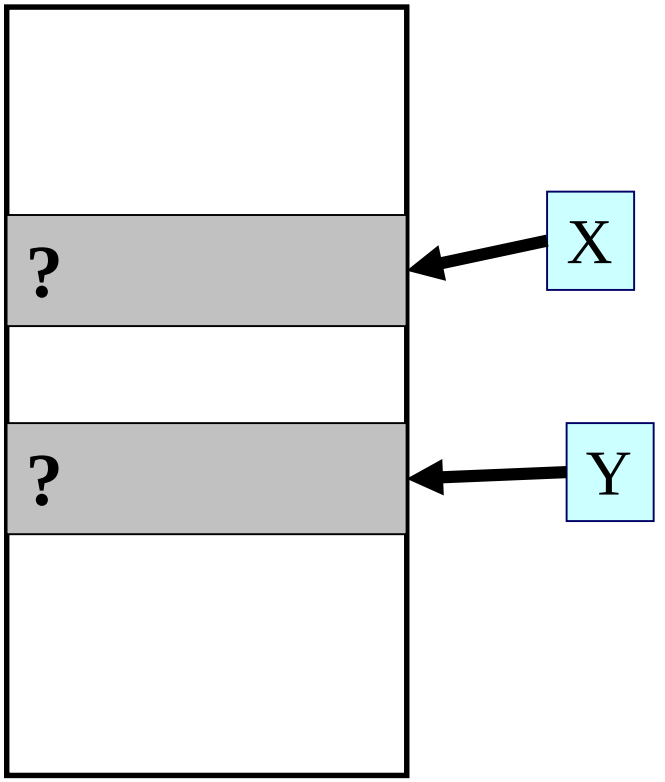
Instruction executed

Time

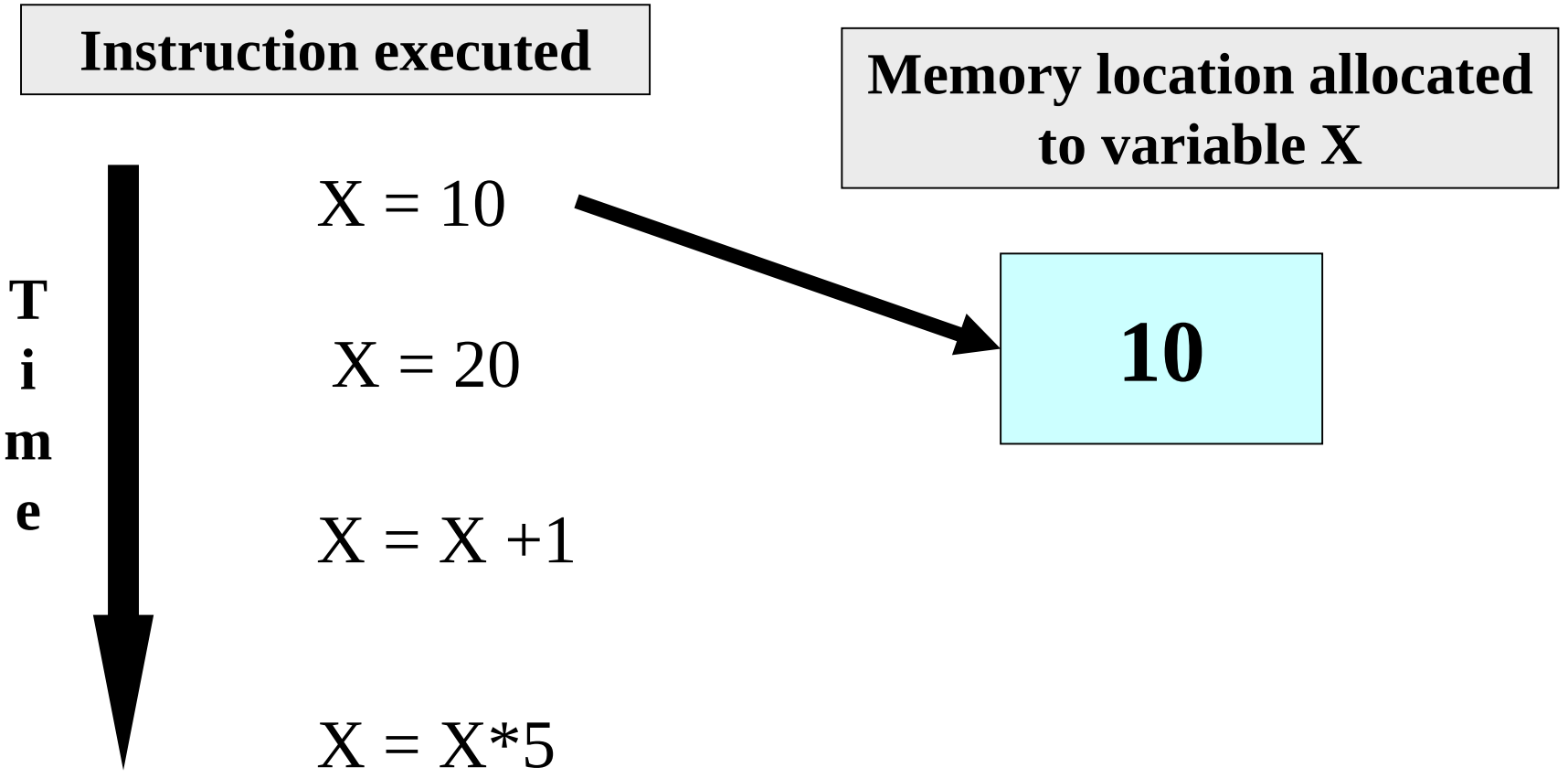
int X

int Y

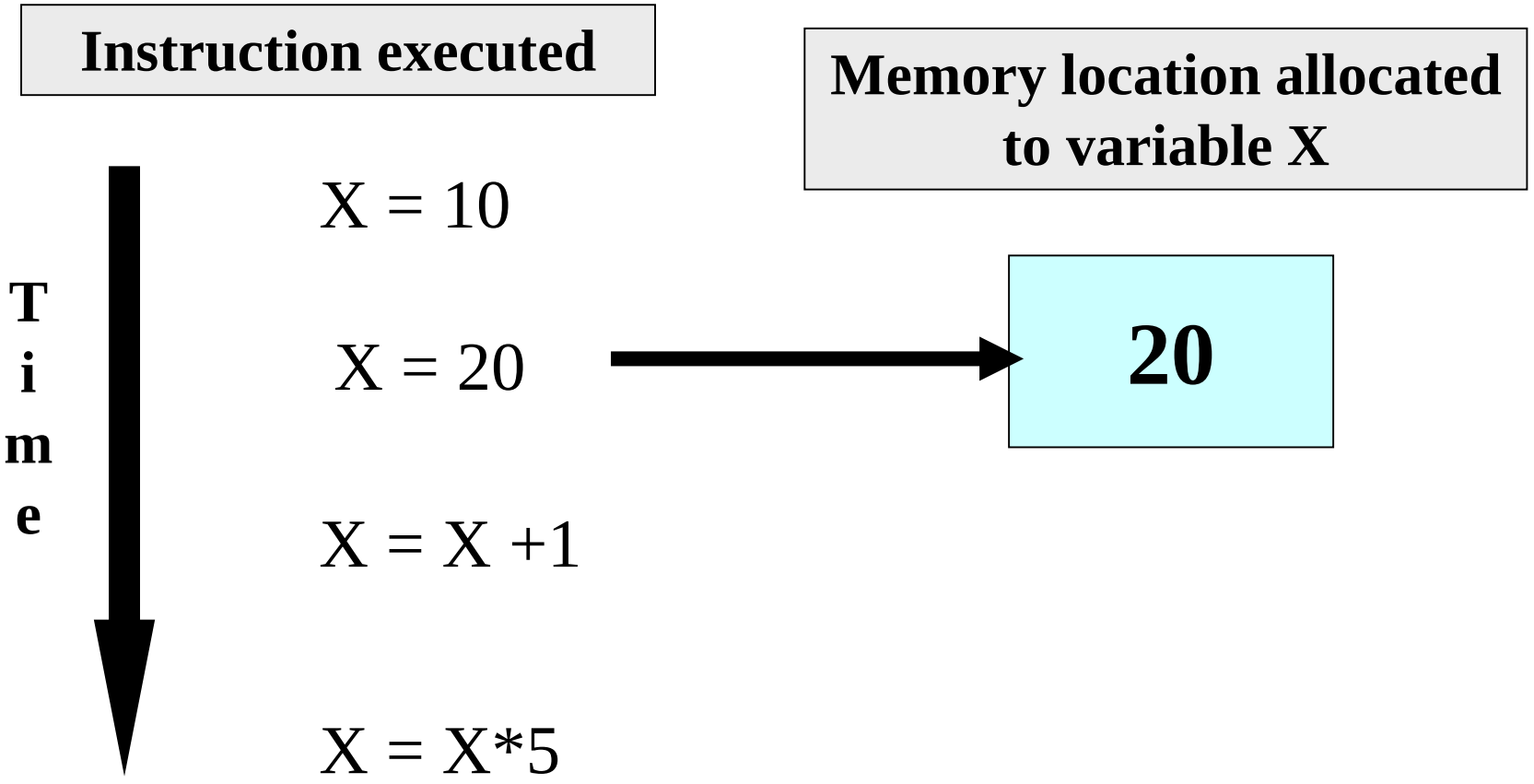
Memory location allocated to variables X and Y



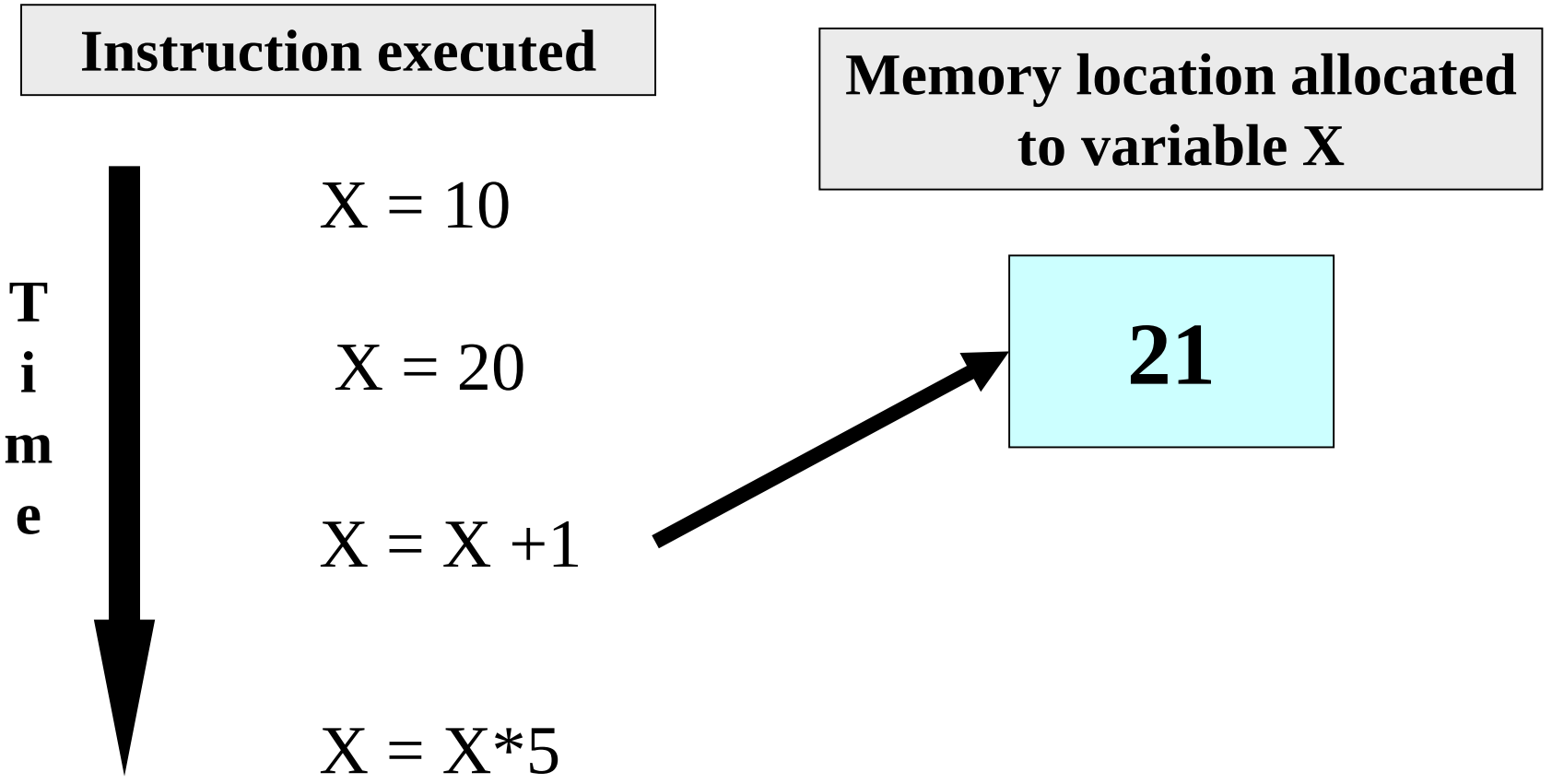
Variables in Memory



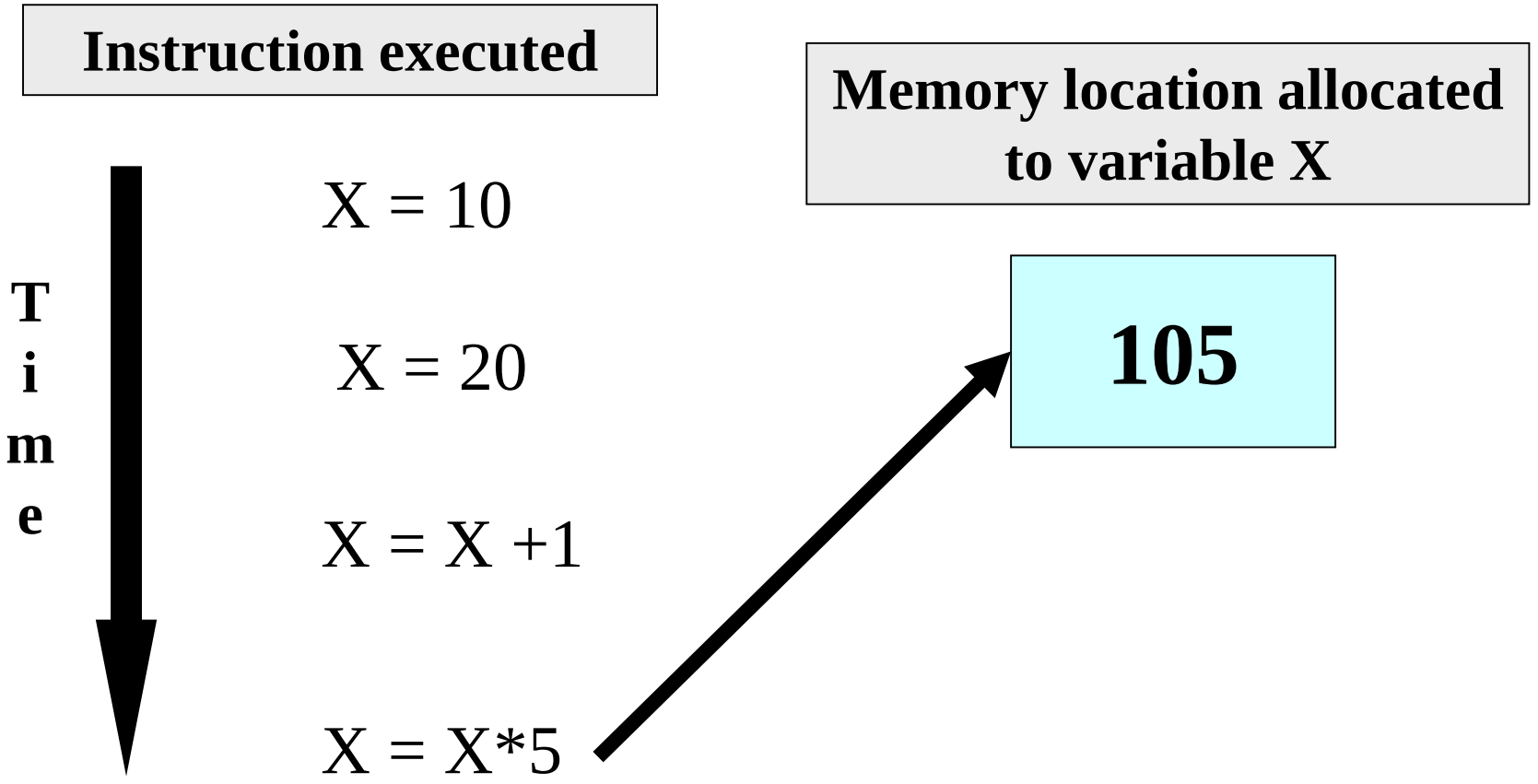
Variables in Memory



Variables in Memory



Variables in Memory

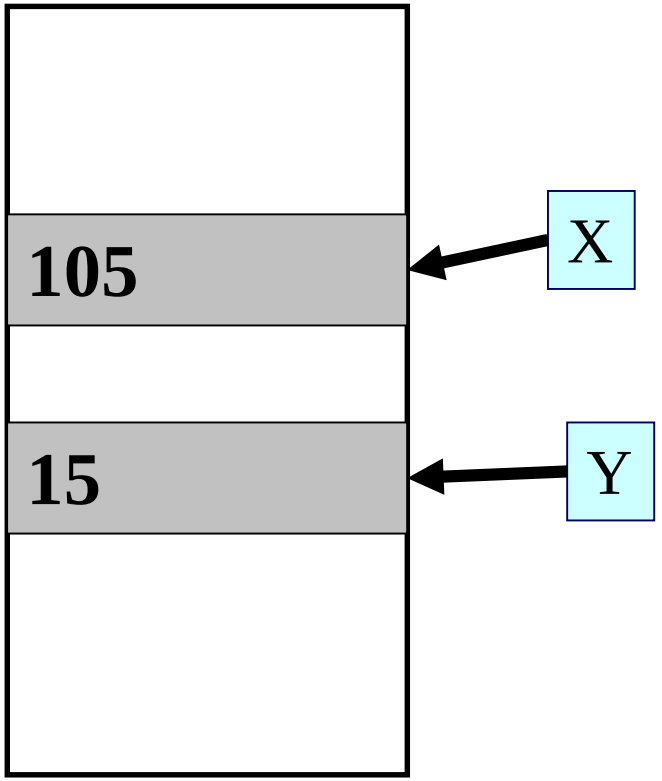


Variables (contd.)

Y=15

Y = Y+3

Y=Y/6

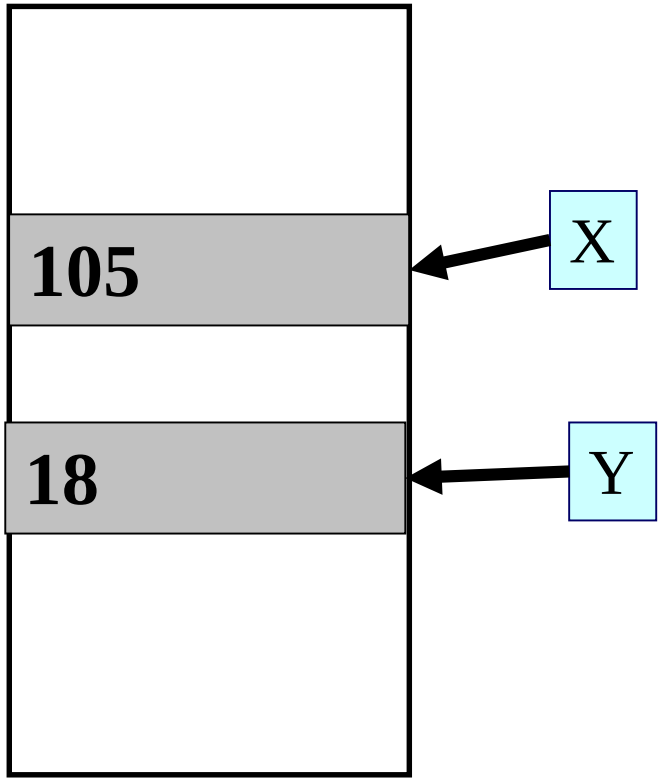


Variables (contd.)

Y=15

Y = Y+3

Y=Y/6

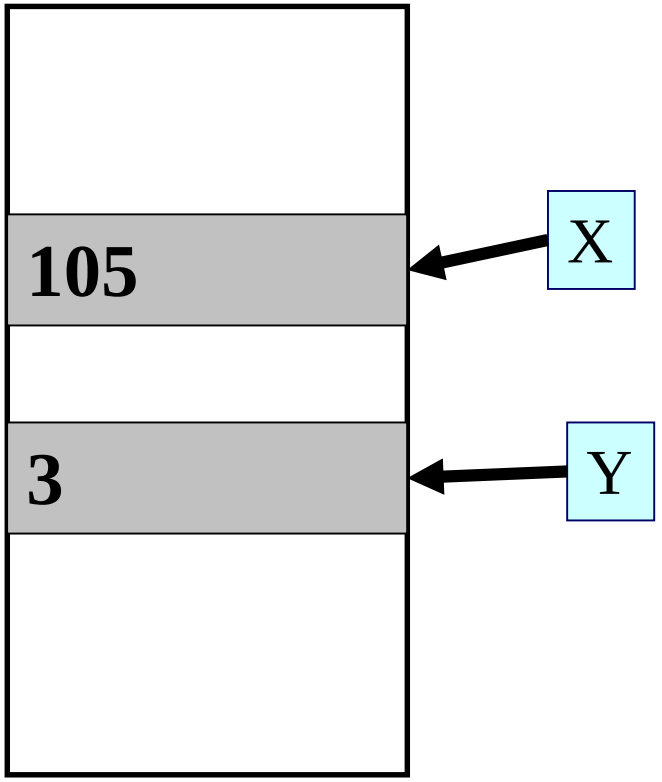


Variables (contd.)

Y=15

Y = Y+3

Y=Y/6



Data Types

- Each variable has a type, indicates what type of values the variable can hold
- Four common data types in C
  - int - can store integers (usually 4 bytes)
  - float - can store single-precision floating point numbers (usually 4 bytes)
  - double - can store double-precision floating point numbers (usually 8 bytes)
  - char - can store a character (1 byte)

Contd.

- First rule of variable use: Must declare a variable (specify its type and name) before using it anywhere in your program
- All variable declarations should ideally be at the beginning of the main() or other functions
  - There are exceptions, we will see later
- A value can also be assigned to a variable at the time the variable is declared

```
int   speed = 30;
char  flag = 'y';
```

Variable Names

- Sequence of letters and digits
- First character must be a letter or '_'
- No special characters other than '_'
- No blank in between
- Names are case-sensitive (max and Max are two different names)
- Examples of valid names:
  - i rank1 MAX max Min class_rank
- Examples of invalid names:
  - a's fact rec 2sqroot class,rank

More Valid and Invalid Identifiers

- Valid identifiers

  X

  abc

  simple_interest

  a123

  LIST

  stud_name

  Empl_1

  Empl_2

  avg_empl_salary

- Invalid identifiers

  10abc

  my-name

  “hello”

  simple interest

  (area)

  %rate

C Keywords

- Used by the C language, cannot be used as variable names
- Examples:
  - int, float, char, double, main, if else, for, while. do, struct, union, typedef, enum, void, return, signed, unsigned, case, break, sizeof,….
  - There are others, see textbook…

Example 1

```
#include <stdio.h>
int main()
{
    int x, y, sum;
    scanf("%d%d",&x,&y);
    sum = x + y;
    printf( "%d plus %d is %d\n", x, y, sum );
    return 0;
}
```

Three int type variables declared

Values assigned

Example - 2

```
#include <stdio.h>
int main()
{
      float x, y;
      int d1, d2 = 10;
      scanf("%f%f%d",&x, &y, &d1);
      printf( "%f plus %f is %f\n", x, y, x+y);
      printf( "%d minus %d is %d\n", d1, d2, d1-d2);
     return 0;
}
```

Assigns an initial value to d2, can be changed later

Read-only variables

- Variables whose values can be initialized during declaration, but cannot be changed after that
- Declared by putting the const keyword in front of the declaration
- Storage allocated just like any variable
- Used for variables whose values need not be changed
  - Prevents accidental change of the value

Correct

```
int main() {
   const int LIMIT = 10;
   int n;
   scanf("%d", &n);
   if (n > LIMIT)
      printf("Out of
    limit");
   return 0;
}
```

Incorrect: Limit changed

```
int main() {
    const int Limit = 10;
    int n;
    scanf("%d", &n);
    Limit = Limit + n;
    printf("New  limit is %d", Limit);
    return 0;
}
```

2-const-change-error.c

Constants

- Integer constants
  - Consists of a sequence of digits, with possibly a plus or a minus sign before it
  - Embedded spaces, commas and non-digit characters are not permitted between digits
- Floating point constants
- Two different notations:
  - Decimal notation: 25.0, 0.0034, .84, –2.234
  - Exponential (scientific) notation

    3.45e23, 0.123e-12, 123e2

e means “10 to the power of”

Contd.

- Character constants
  - Contains a single character enclosed within a pair of single quote marks.
  - Examples :: ‘2’, ‘+’, ‘Z’
- Some special backslash characters

‘\n’	new line
‘\t’	horizontal tab
‘\’’	single quote
‘\”’	double quote
‘\\’	backslash
‘\0’	null

Typical Size of Data Types

- char – 1 byte
- int – 4 bytes
- float – 4 bytes
- double – 8 bytes

- “Typical”, because some of them vary depending on machine/OS type
- Never use the values (1, 4, 8) directly, use the sizeof() operator given
  - sizeof(char) will give 1, sizeof(int) will give 4 and so on your PC/Laptop

Input: scanf function

- Performs input from keyboard
- It requires a format string and a list of variables into which the value received from the keyboard will be stored
- format string = individual groups of characters (usually '%' sign, followed by a conversion character), with one character group for each variable in the list

```
int a, b;
float c;
scanf("%d %d %f", &a, &b, &c);
```

Variable list (note the & before a variable name)

Format string

- Commonly used conversion characters
  - c for char type variable
  - d for int type variable
  - f for float type variable
  - lf for double type variable

- Examples

```
scanf ("%d", &size) ;
scanf ("%c", &nextchar) ;
scanf ("%f", &length) ;
scanf (“%d%d”, &a, &b);
```

Examples

- scanf ("%d", &size) ;
  - Reads one integer from keyboard into an int type variable named size
- scanf ("%c", &nextchar) ;
  - Reads one character from keyboard into a char type variable named nextchar

Examples

- scanf ("%f", &length) ;
  - Reads one floating point (real) number from keyboard into a float type variable named length
- scanf (“%d%d”, &a, &b);
  - Reads two integers from keyboard, the first one in an int type variable named a and the second one in an int type variable named b

- Important:
  - scanf will wait for you to type the input from the keyboard
  - You must type the same number of inputs as the number of %'s in the format string
  - Example: if you have scanf("%d%d",…), then you must type two integers (in same line or different lines), or scanf will just wait and the next statement will not be executed

Reading a single character

- A single character can be read using scanf with %c
- It can also be read using the getchar() function

```
char c;
c = getchar();
```

- Program waits at the getchar() line until a character is typed, and then reads it and stores it in c

Output: printf function

- Performs output to the standard output device (typically defined to be the screen)
- It requires a format string in which we can specify:
  - The text to be printed out
  - Specifications on how to print the values

```
printf ("The number is %d\n", num);
```

  - The format specification %d causes the value listed after the format string to be embedded in the output as a decimal number in place of %d
  - Output will appear as: The number is 125

Contd.

- General syntax:

  printf (format string, arg1, arg2, …, argn);

  - format string refers to a string containing formatting information and data types of the arguments to be output
  - the arguments arg1, arg2, … represent list of variables/expressions whose values are to be printed
- The conversion characters are the same as in scanf

- Examples:

```
printf  (“Average of %d and %d is %f”, a, b, avg);
printf  (“Hello \nGood \nMorning \n”);
printf (“%3d %3d %5d”, a, b, a*b+2);
printf (“%7.2f  %5.1f”, x, y);
```

- Many more options are available for both printf and scanf
  - Read from the book
  - Practice them in the lab

More Examples

(Explain the outputs to test if you understood format strings etc.)

More print

```
#include <stdio.h>
int main()
{
    printf ("Hello, World! ") ;
    printf ("Hello \n World! \n") ;
    return 0;
}
```

Output

```
Hello, World! Hello
 World!
```

Some more print

```
#include <stdio.h>
int main()
{
    printf ("Hello, World! \n") ;
    printf ("Hello \n World! \n") ;
    printf ("Hell\no \t World! \n") ;
    return 0;
}
```

Output

```
Hello, World!
Hello
 World!
Hell
o       World!
```

Some more print

```
#include <stdio.h>
int main()
{
    float f1, f2;
    int x1, x2;
    printf("Enter values for f1 and f2: \n");
    scanf("%f%f", &f1, &f2);
   printf("Enter values for x1 and x2: \n");
    scanf("%d%d", &x1, &x2);
   printf("f1 = %f, f2 = %5.2f\n", f1, f2);
    printf("x1 = %d, x2 = %10d\n", x1, x2);
     return 0;
}
```

Output

```
Enter values for f1 and f2:
23.5  14.326
Enter values for x1 and x2:
54  7
f1 = 23.500000, f2 =      14.33
x1 = 54, x2 =          7
```

Can you explain why 14.326 got printed as 14.33?

Some more print

```
#include <stdio.h>
int main()
{
   char c1, c2;
   scanf("%c%c", &c1, &c2);
   printf("%c%c", c1, c2);
    return 0;
}
```

Output

```
ab
ab
```

What about this?

```
#include <stdio.h>
int main()
{
   char c1, c2;
   scanf("%c%c", &c1, &c2);
   printf("%c%c", c1, c2);
    return 0;
}
```

Output

```
a b
a
```

Can you explain why only 'a' was printed this time, even though it is the same program as in the last slide? Note the difference from the last slide carefully. Also note that two characters were read this time also, or scanf would have waited

Practice Problems

- Write C programs to
  1. Read two integers and two floating point numbers, each in a separate scanf() statement (so 4 scanf's) and print them with separate printf statements (4 printf's) with some nice message
  2. Repeat 1, but now read all of them in a single scanf statement and print them in a single printf statement
  3. Repeat 1 and 2 with other data types like double and char
  4. Repeat 1 and 2, but now print all real numbers with only 3 digits after the decimal point
  5. Read 4 integers in a single scanf statement, and print them (using a single printf statement) in separate lines such that the last digit of each integer is exactly 10 spaces away from the beginning of the line it is printed in (the 9 spaces before will be occupied by blanks or other digits of the integer). Remember that different integers can have different number of digits
  6. Repeat 5, but now the first integer of each integer should be exactly 8 spaces away from the beginning of the line it is printed in.