Data

- int

Expressions

- I/O expressions
- Arithmetic expressions

Control Flow

- Sequential

Data

- int
- float
- double

Expressions

- I/O expressions
- Arithmetic expressions

Control Flow

- Sequential

`float` and `double` Data Types

`float` and `double` Data Types

Kind of data:

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)

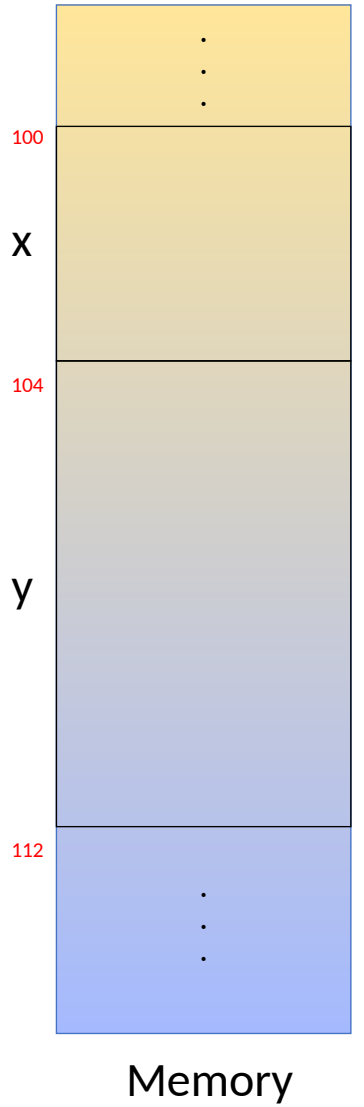
`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

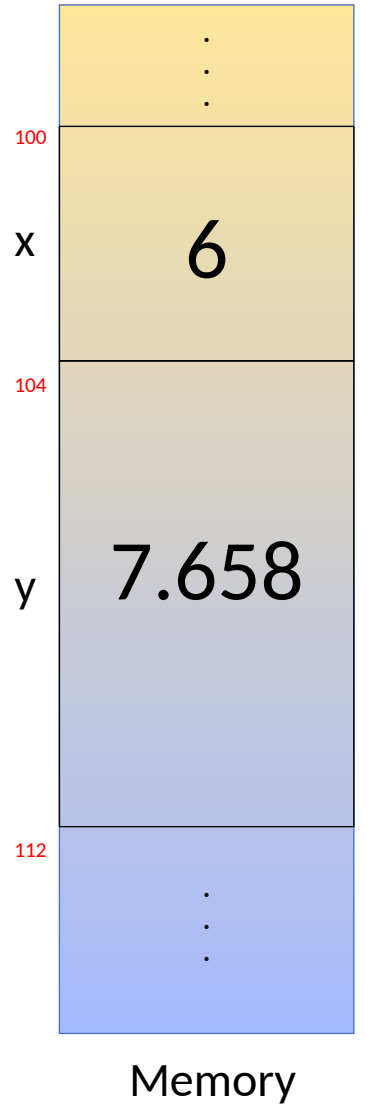
Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)

```
int main(){
        int x;
        double y;



        return 0;
}
```



```
int main(){
        int x;
        double y;

        x = 6;
        y = 7.658;

        return 0;
}
```



`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)

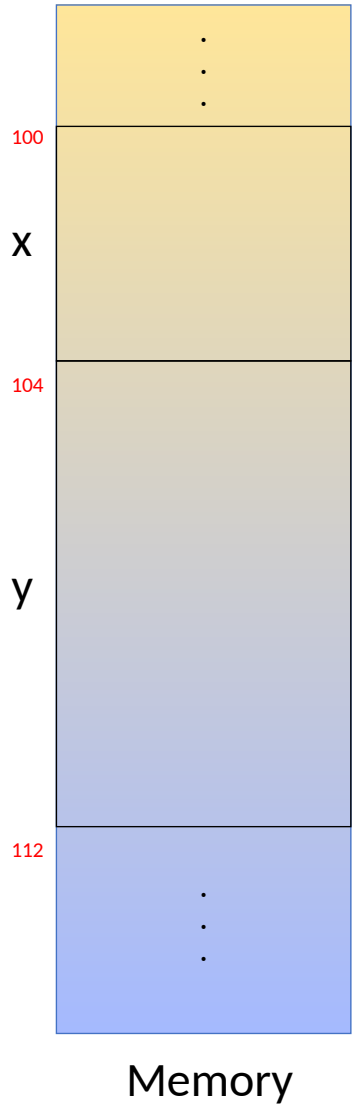
`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

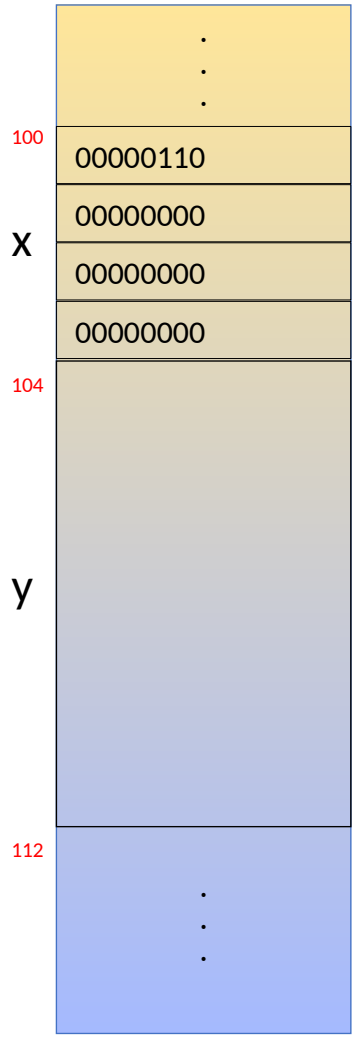
Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)
- The numbers are represented by the floating point method (IEEE-754)

```
int main(){
        int x;
        double y;

        x = 6;
        y = 7.658;

        return 0;
}
```

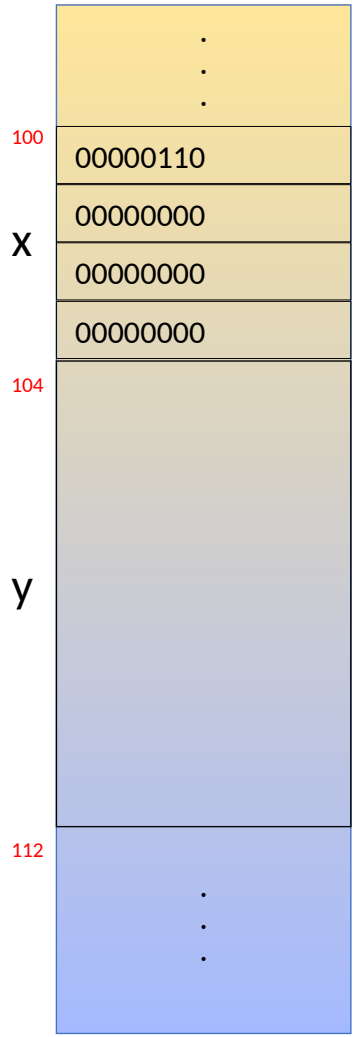


```
int main(){
        int x;
        double y;

        x = 6;
        y = 7.658;

        return 0;
}
```



$(6)_{10} = (00000000\ 00000000\ 00000000\ 00000110)_{2's}$

```
int main(){
        int x;
        double y;

        x = 6;
        y = 7.658;

        return 0;
}
```

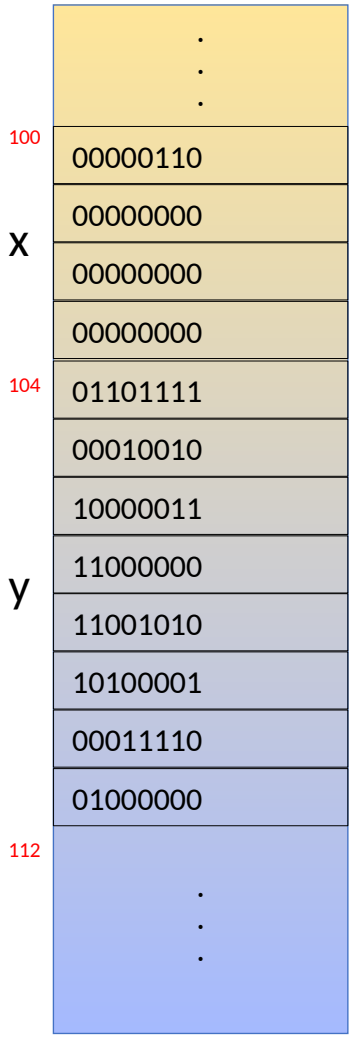


$(6)_{10}$ = $(00000000\ 00000000\ 00000000\ 00000110)_{2's}$

$(7.658)_{10}$ =

= $(01000000\ 00011110\ 10100001\ 11001010\ 11000000\ 10000011\ 00010010\ 01101111)_{IEEE\text{-}754}$

```
int main(){
        int x;
        double y;

        x = 6;
        y = 7.658;

        return 0;
}
```



$(6)_{10}$ = $(00000000\ 00000000\ 00000000\ 00000110)_{2's}$

$(7.658)_{10}$ =

= $(01000000\ 00011110\ 10100001\ 11001010\ 11000000\ 10000011\ 00010010\ 01101111)_{IEEE\text{-}754}$

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)
- The numbers are represented by the floating point method (IEEE-754)

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)
- The numbers are represented by the floating point method (IEEE-754)

C++ literals:

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)
- The numbers are represented by the floating point method (IEEE-754)

C++ literals:

For `double`: 3.4, -8.975, 6.0, …

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)
- The numbers are represented by the floating point method (IEEE-754)

C++ literals:

For `double`: 3.4, -8.975, 6.0, …

For `float`: 3.4f, -8.975f, …

`float` and `double` Data Types

Kind of data: Real numbers (could have fractional part)

Inner representation:

- `double` – Each data uses 8 bytes (64 bits)
  `float` – Each data uses 4 bytes (32 bits)
- The numbers are represented by the floating point method (IEEE-754)

C++ literals:

For `double`: 3.4, -8.975, 6.0, …

For `float`: 3.4f, -8.975f, …

Arithmetic Operators: +, -, *, /, =, …

Area of a Circle

Write a program that reads from the user a radius of a circle.
The program will then print area of this circle.

Area of a Circle

Write a program that reads from the user a radius of a circle.
The program will then print area of this circle.

Example

Please enter the radius:

Area of a Circle

Write a program that reads from the user a radius of a circle.
The program will then print area of this circle.

Example

Please enter the radius:

2.6

Area of a Circle

Write a program that reads from the user a radius of a circle.
The program will then print area of this circle.

Example

Please enter the radius:

2.6

The area of a circle with radius of 2.6 is 21.2372

Area of a Circle

Area of a Circle

Theorem:

Let $\mathbf{C}$ be a circle with a radius of length $\mathbf{r}$.
We have:

(Area of $\mathbf{C}$) = $\pi \times \mathbf{r}^2$

Type Casting

```
int main(){
        int x1, x2;
        double y1, y2;









        return 0;
}
```

Type Casting

```
int main(){
        int x1, x2;
        double y1, y2;

        x1 = 6;
        y1 = 6.7;




        return 0;
}
```

Type Casting

```
int main(){
        int x1, x2;
        double y1, y2;

        x1 = 6;
        y1 = 6.7;

        y2 = 6;


        return 0;
}
```

Type Casting

```
int main(){
        int x1, x2;
        double y1, y2;

        x1 = 6;
        y1 = 6.7;

        y2 = 6;


        return 0;
}
```

Type Casting

```
int main(){
        int x1, x2;
        double y1, y2;

        x1 = 6;
        y1 = 6.7;

        y2 = (double)6;


        return 0;
}
```

Casting

converting the representation of a data from one type to another type

Type Casting

```
int main(){
        int x1, x2;
        double y1, y2;

        x1 = 6;
        y1 = 6.7;

        y2 = (double)6;
        x2 = (int)6.7;

        return 0;
}
```

Casting

converting the representation of a data from one type to another type

Expressions With Mixed Types

```
int main(){
        int x;
        double y;









        return 0;
}
```

Expressions With Mixed Types

```
int main(){
        int x;
        double y;

        cout<< 5 / 2 <<endl;
        cout<< 5.0 / 2.0
<<endl;




        return 0;
}
```

Expressions With Mixed Types

```
int main(){
	int x;
	double y;

	cout<< 5 / 2 <<endl;
	cout<< 5.0 / 2.0
<<endl;
	cout<< 5.0 / 2 <<endl;




	return 0;
}
```

Expressions With Mixed Types

```
int main(){
	int x;
	double y;

	cout<< 5 / 2 <<endl;
	cout<< 5.0 / 2.0
<<endl;
	cout<< 5.0 / 2 <<endl;

	x = 5/2;


	return 0;
}
```

Expressions With Mixed Types

```
int main(){
	int x;
	double y;

	cout<< 5 / 2 <<endl;
	cout<< 5.0 / 2.0
<<endl;
	cout<< 5.0 / 2 <<endl;

	x = 5/2;
	y = 5/2;

	return 0;
}
```